Adaptive Agent-Based Intrusion Response

TEXAS A&M UNIVERSITY

1876

Curtis A. Carver, Jr.

ADAPTIVE AGENT-BASED INTRUSION RESPONSE

A Dissertation

by

CURTIS A. CARVER JR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2001

Major Subject: Computer Science

ADAPTIVE AGENT-BASED INTRUSION RESPONSE

A Dissertation

by

CURTIS A. CARVER JR

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Udo W Pooch

Udo W. Pooch
(Chair of Committee)

John

John Yen
(Member)

Michael Longnecker

Michael T. Longnecker
(Member)

Jyh-Chen S. Liu

Steve Liu
(Member)

Bart Childs

Wei Zhao
(Head of Department)

May 2001

Major Subject: Computer Science

ABSTRACT

Adaptive Agent-Based Intrusion Response. (May 2001)

Curtis A. Carver Jr, B.S., United States Military Academy;

M.C.S., Texas A&M University

Chair of Advisory Committee: Dr. Udo W. Pooch

A new methodology has been developed for adaptive, automated intrusion response (IR) focusing on the role of software agents in providing that response. The majority of intrusion response systems (IRSs) react to attacks by generating reports or alarms. This introduces a window of vulnerability between when an intrusion is detected and when action is taken to defend against the attack. This window of vulnerability has been reduced through an agent-based system that adaptively responds to intrusions.

Multiple IDSs monitor a computer system and generate intrusion alarms. Interface agents maintain a model of each IDS based on the number of false positives/negatives previously generated. It uses this model to generate an attack confidence metric and passes this metric along with the intrusion alarm to the Master Analysis agent. The Master Analysis agent classifies whether the incident is a continuation of an existing incident or is a new attack. If it is a new attack, the Master Analysis agent creates a new Analysis agent to develop a response plan to the new attack. If the incident is a continuation of an existing attack, the Master Analysis agent passes the attack confidence metric and intrusion alarm to the existing Analysis agent handling the attack. The Analysis agent analyzes an incident until it is resolved and generates a course of action to resolve the incident. To generate this course of action, the Analysis agent involves the Response Taxonomy agent to classify the attack and Policy Specification agent to limit the response based on legal, ethical, institutional, or resource constraints. The Analysis agent creates a course of action and then invokes the appropriate components of the Response Toolkit. The Analysis agents employ adaptive decision-making based on the success of previous responses. As decisions are made, the results are displayed to the user interface.

This research presents a novel IR methodology that includes: response adaptation to intrusive behavior based on confidence in the intrusion detection mechanism; response adaptation to intrusive behavior based on the success of previous intrusion responses; and, synergistic support for multiple IDSs.

DEDICATION

This dissertation is dedicated to my wife, Eileen, and my children, Curtis, Gregory and Michelle. My joy in life is often simply a pale reflection of the love in their eyes.

ACKNOWLEDGMENTS

This research would not have been possible without the support of my mentors, family and colleagues. I would like to thank my chair Dr. Udo Pooch for his guidance and support during my graduate education. As an army officer selected to return to graduate school, I could have attended almost any university in the United States. After obtaining my master's degree under Dr. Pooch's tutelage nearly ten years ago, it was an easy choice for me to return to Texas A&M and to Dr. Pooch. Although he is not the easiest person to work for, there is no one else that I would rather have as my chair or as a mentor.

The other members of my committee, Drs. Yen, Liu, Longnecker, and Wood significantly strengthened this research through their guidance and support. Rather than simply show up to evaluate my work at my proposal and dissertation defense, my committee members actively guided me throughout this research. Mainly through short one-on-one sessions, my committee members helped me transform the concept of active intrusion response into a comprehensive methodology and functional prototype. For this unselfish and professional support, I am very graceful.

My success for the last twenty-one years has been in large part due to my lovely wife Eileen. She is my best friend and has always been there for me, regardless of the task. She kept the household running smoothly and efficiently despite the assistance of our three overly happy and enthusiastic children. This allowed me the freedom to conduct this research. There would be no PhD without her support.

Finally, I would like to acknowledge the help I received from my friends John Hill, Buck Surdu, Jeff Humphries, Willis Marti, Dan Ragsdale, Dave Hess, Ellen Mitchell and Dr. Tom Ioerger. The long discussions, brainstorming sessions, and sometimes even heated debates helped me develop and refine many of the ideas in this research. This is a better dissertation due to their input and I am a better person for having such good friends.

TABLE OF CONTENTS

CHAPTER Page

Page

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Motivation

The number of information warfare attacks is increasing and becoming increasingly sophisticated. Annual reports from the Computer Emergency Response Team (CERT) indicate a significant increase in the number of computer security incidents each year. Figure 1 depicts the rise of computer security incidents with six incidents reported in the 1988 and 8,268 in 1999 [1]. Not only are these attacks becoming more numerous, they are also becoming more sophisticated. The 1998 CERT Annual Report reports the growing use of "widespread attacks using scripted tools to control a collection of information-gathering and exploitation tools" [2]. The 1999 CERT Distributed Denial of Service Workshop likewise reports the growing use of automated scripts that launch and control tens of thousands of attacks against one or more targets. Each attacking computer has limited information on who is initiating the attack and from where [3]. The threat of sophisticated computer attacks is growing. Unfortunately, intrusion detection and response systems have not kept up with the increasing threat.

IEEE Transactions on Automatic Control is used as a pattern for format and style.

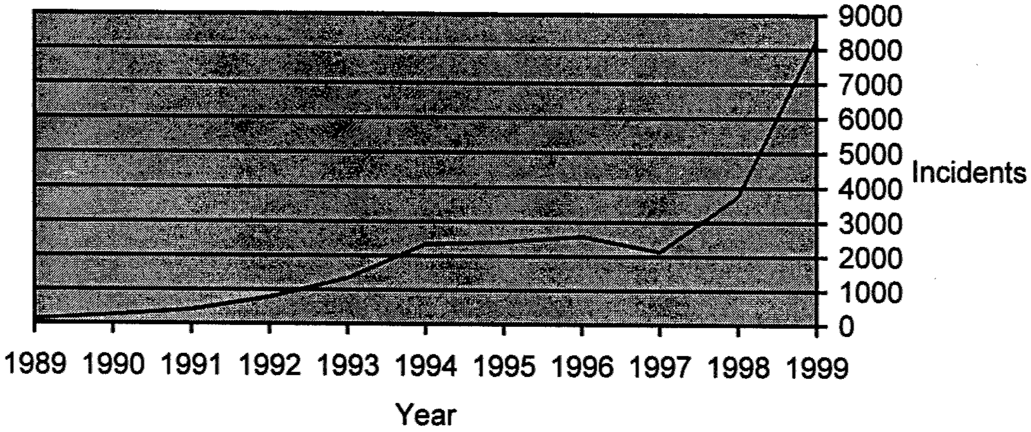


Figure 1: Number of CERT Reported Incidents per Year

Current intrusion detection systems (IDS) have limited response mechanisms that are inadequate given the current threat. While IDS research has focused on better techniques for intrusion detection, intrusion response remains principally a manual process. The IDS notifies the system administrator that an intrusion has occurred or is occurring and the system administrator must respond to the intrusion. Regardless of the notification mechanism employed, there is a delay between detection of a possible intrusion and response to that intrusion.

This delay in notification and response, ranging from minutes to months, provides a window of opportunity for attackers to exploit. Cohen explored the effect of reaction time on the success rate of attacks using simulations [4]. The results indicate that if skilled attackers are given ten hours after they are detected before a response, they will be successful 80% of the time. If they are given twenty hours, they will succeed 95% of the time. At thirty hours, the attacker almost never fails. The simulation results were also correlated against the skill of the defending system administrator. The results indicate that if a skilled attacker is given more than thirty hours, the skill of the system administrator becomes irrelevant - the attacker will succeed. On the other hand, if the response is instantaneous, the probability of a successful attack against a skilled system administrator is almost zero. Response is a fundamental factor in whether or not an attack is successful. An automated intrusion response system provides the best possible defense and shortens or closes this window of opportunity until the system administrator can take an active role in defending against the attack. Unfortunately, no such response system exists.

B. Research Objectives

The overall intent of this research is to develop an intrusion response taxonomy and a methodology for automatic offensive and defensive response. This methodology will support the automatic defense of computer systems through an autonomous, agent-based adaptive architecture. The following research objectives will accomplish this intent:

- Research and develop a taxonomy of intrusion responses based on:
  - Type of attack using the Confidentially/Integrity/Availability (CIA) model for classification of attack.
  - Type of attacker.
  - Timing of response.
  - Implications of the attack.

  - Strength of suspicion.
  - Environmental constraints (legal, ethical, resource-based).
- Develop a methodology for responding to an intrusion that includes:
  - A hierarchical organization of software agents, including:
    - Master analysis agents that correlate incidents with ongoing attacks and pass the incident to the responsible analysis agent or launch new analysis agents when new attacks occur.
    - Analysis agents that analyze an attack over the life of the attack and develop a course of action to response to the attack. Once a course of action is determined, the analysis agent launches the appropriate response toolkit agents (simulated).
  - A formal reasoning mechanism to provide for adaptation of the response.
- Develop a prototype system to demonstrate the feasibility of this approach.

C. Overview

Chapter II presents a review of the current literature relevant to the domains involved in this research. The purpose of Chapter II is to describe work that has been done to date and to provide background and motivational material for this research. Domain areas of interest to this dissertation are IDS, intrusion response systems (IRS), security taxonomies, and artificial intelligence (specifically expert systems and software agents).

Chapter III describes the proposed methodology for responding automatically to computer attacks. This chapter describes the various components of this methodology, paying particular attention to those components that have been prototyped.

Chapter IV gives details of the implemented prototype - the Adaptive, Agent-based Intrusion Response system (AAIR). It describes the interactions of these prototype components and the performance of the overall system.

Chapter V describes the verification and validation procedures used to confirm the correctness and effectiveness of the prototype. Chapter V also describes some experiments conducted to demonstrate that the software agents perform as designed (verification). Finally, Chapter V describes the results of this research.

Chapter VI presents the major conclusions of this research, reasserts the contributions of this research to the field, and makes recommendations for future research.

CHAPTER II

LITERATURE REVIEW

A. Introduction

The development of an intrusion response system requires the discussion of a number of different domains including intrusion detection systems, intrusion response systems, security taxonomies, and artificial intelligence (specifically software agents). Intrusion response systems (IRS) are dependent on intrusion detection systems (IDS) in two respects: (1) IDS detect the intrusions that the IRS must respond to and, (2) IDS are imperfect which requires IRS to adapt the response based on its confidence in the detection capabilities of the IDS. There has been previous research in IRS as all IDS contain some intrusion response component ranging from report generation to automatic defense of the system. Unfortunately, intrusion response has rarely been discussed by itself. Instead, most research has focused on the detection of intrusions with intrusion response being left as the responsibility of the system administrator. As a result, the intrusion response mechanisms within these systems are limited. Similarly, security taxonomy research has focused on the development of security flaw taxonomies and not security response taxonomies. A response taxonomy provides the theoretical framework for classifying responses and as such is an important component of this research. Finally, while there is a rich body of research on artificial intelligent and software agents, this research has not been applied to the problem of automated intrusion response. Each of these domains will be discussed in sufficient detail to provide a background for this research.

B. Intrusion Detection Systems (IDS)

1. Historical Perspective

Anderson introduced the concept of intrusion detection in 1980. He defined an intrusion as "an unauthorized attempt to access or manipulate information, or to render a system unreliable or unstable" [5]. His paper went on to define several terms in computer security and classify six categories of intrusive activities and how these activities might be detected: attempted break-ins, masquerade attacks, penetration of the security control system, leakage, and denial of service. The detection mechanisms recommended included monitoring unusual behavior profiles, uncommon uses of system resources, and monitoring for specific patterns of activity [5]. These recommendations led to the development of two of the three principal approaches, anomaly detection and misuse detection in intrusion detection systems.

Anderson also created a taxonomy of system intruders whom he divided into internal and external intruders. Internal users are further divided into masqueraders, misfeasors, and clandestine users. Masqueraders are attackers that exploit user accounts and associated privileges. Misfeasors are legitimate users that use their privileges to participate in illicit activity. Finally, clandestine users are attackers that gain supervisory control of the system [5]. Anderson introduced the concepts and terminology that provided the early theoretical foundations for IDS.

Denning extended Anderson's work in 1987 through the introduction of a generic intrusion detection model [6]. Denning's model is composed of six components: subjects, objects, audit records, profiles, anomaly records, and activity rules. Subjects are the initiators of activity and each subject has an associated profile that characterizes that subject's behavior. Subjects utilize objects which are system-managed resources. The use of these resources generates audit records, which can be compared against subject profiles. If there is a significant deviation between the audit record and subject profile, the system generates anomaly records. The activity rules contain the rules used to determine what action to execute when the system generates an audit or anomaly record, or a time period ends [6].

While Denning focused on a generic model, she also provided a broad framework for future intrusion detection research. Anomaly detection is discussed in detail with a number of metrics and statistical models for evaluating these metrics. Misuse detection is introduced and some of the disadvantages with misuse detection are discussed in the context of why misuse detection was not included in the Intrusion Detection Expert System (IDES). Denning's work spurred interest in intrusion detection from which a variety of IDSs have been developed.

There are three broad approaches for intrusion detection: anomaly detection, misuse detection, and specification-based detection. In practice, none of the three are sufficient for a robust intrusion detection system - a combination of two or all three approaches is necessary. The characteristics and limitations of these approaches are discussed below.

2. Anomaly Detection

Anomaly detection is based on the premise that intrusions are a subset of anomalous activity. Anomaly detection IDSs monitor user activity and report significant deviations from normal activity as intrusions. Monitoring can be at a system or user level and consists of comparing activity against a user profile. The user profile is a collection of metrics such as average CPU load, number of processes, login time, or number of network connections that characterizes user activity. Threshold levels are set for these metrics, and activity above these thresholds are characterized as intrusions [6].

Because intrusions are a subset of anomalous activity, it is possible to flag an anomalous activity as intrusive when it is not (*false positive*), or to ignore intrusive behavior because the anomaly detection system does not consider it abnormal (*false negative*).

There are a number of compromises involved in building anomaly detection systems. The effectiveness of the system is dependent on the number of metrics monitored and the frequency at which these metrics are monitored. The accuracy of the anomaly detection increases as the number of metrics and frequency of monitoring increases. The system requirements of the anomaly detection system likewise increase requiring a compromise between system performance and model accuracy.

3. Misuse Detection

Misuse detection is based on the premise that all intrusions have a distinct signature that can be detected. Misuse detection systems maintain a collection of attack signatures and monitor the system for an attack. If user or system activity matches a signature, then the system reports an intrusion.

Misuse detection systems can report false positives and negatives like anomaly-based systems. If a signature matches normal user activity as well as intrusive behavior, then a false positive is reported. If a new attack is developed for which an attack signature does not exist, then a false negative will occur.

4. Specification-based Detection

Specification-based detection focuses on expected system behavior instead of user activity. System behavior is formally specified for all circumstances and a profile is developed. The system is then monitored and all its actions are compared against the profile; system behavior that is not specified as correct is flagged as an intrusion [7].

A possible implementation of specification-based detection system is the use of a special policy specification language. This specification language would stipulate security policy by assigning access privileges to each file in the system.

Specification-based detection systems can have false negatives but if system behavior is specified accurately, there are no false positives. False negatives can occur when the system specification does not cover all possible system states. False positives can only occur if the system behavior is not specified accurately.

Table 1: Comparison of IDS Approaches

Approach	Advantages	Disadvantages
Anomaly	Can detect new attacks without reprogramming. Few false negatives.	Potential for many false positives. Insiders can train user model to classify intrusive behavior as normal.
Misuse	Few false positives	Potential for many false negatives due to vulnerabilities to unknown attacks. Easy to mask attack
Specification	Potentially no false positives	Very difficult to specify all system states.

5. Comparison of IDS Approaches

IDS approaches address different types of intruders. Anomaly systems detect marauders better than misuse systems under the assumption that the marauder's usage pattern is significantly different from the user. Misuse systems can detect misfeasors while anomaly systems are generally ineffective because misfeasors can train the anomaly detection system to consider intrusive behavior as "normal" for the user over time. Both anomaly and misuse have limited utility against a clandestine attacker. Once an intruder has supervisory permission on a system, detection becomes very difficult as the skilled clandestine attacker can alter all logging and audit mechanisms to cover his intrusion. No current IDS approach is sufficient for detecting all intrusions. Instead, a combination of approaches is necessary to protect against different types of attacks (See Table 1.

Patterns of usage also influence the effectiveness of a particular IDS approach. If the users are in a production environment where they repeatedly use a limited subset of commands in a particular order, anomaly detections work extremely well. If the users use the system infrequently or have no set pattern of usage, then misuse detection systems tend to outperform anomaly detection systems.

Table 1 summarizes the advantages and disadvantages of each intrusion detection approach. Most IDS implement a combination of approaches to balance the advantages and disadvantages of each approach.

6. IDS Classification Techniques

There are a number of classification techniques that can be used within intrusion detection approaches. These techniques classify events as either intrusive or normal and include statistical analysis, predictive patterns, state transition, expert systems, neural networks, machine learning, pattern matching, graph-based, and model-based approaches. This section will examine these techniques.

a) Statistical Analysis: Statistical analysis is an anomaly detection technique that uses differences in the volume and type of audit data to detect intrusions. This is one of the earliest forms of intrusion detection and has been used in a large number of IDSs. There are three forms of statistical analysis used for intrusion detection: threshold detection, profile-based, and keystroke monitoring [8].

Threshold detection uses summary statistics on system and user activities to detect intrusions. The parameters of a threshold detection system are: what activity should the IDS measure and monitor; how often should the IDS perform analysis on this measurement; and what level of activity is considered intrusive. As the first two parameters are increased, the system resources required of the threshold detection increases. The third parameter, the threshold level, depends on the relevance of the security event being monitored and directly affects the number of false positives and false negatives reported by the system. As the threshold is lowered, the probability of false positives increases and false negatives decreases. As the threshold is raised, the converse occurs and the probability of false positives decreases and the false negatives increases [8].

Profile-based detection is based on establishing patterns of normal behavior for a user or system and then classifying significantly deviant behavior as intrusive. It differs from threshold detection in that it employs patterns of usage instead of summary statistics to determine if an intrusion has occurred. The patterns maintained by the IDS are adaptive in that they change over time to reflect the usage patterns of each user accurately [8].

Keystroke monitoring is a misuse detection technique that monitors sequences of keystrokes for attack patterns. This is a very simplistic technique that can be easily evaded through the use of user-defined aliases or the running of intrusive programs that require non-intrusive keystroke entries [9]. While this technique was used in earlier systems, it is seldom used in modern IDS.

b) Artificial Intelligence Techniques: Artificial intelligence techniques are the most commonly used techniques for classifying intrusive behavior. These techniques are also one of the earliest forms of intrusion detection and has been used in almost every IDS. There are four principal artificial intelligence techniques used for intrusion detection: expert systems, predictive patterns, neural networks, and machine learning.

Expert systems have been and continue to be the most popular intrusion detection techniques employed. Expert systems use rules in anomaly or misuse systems to detect attacks. In anomaly detection systems, the rules specify usage patterns based on selected user metrics. In misuse detection systems, the rules stipulate specific types of known attacks. Expert system rules are typically implemented as a series of if-then statements. The principal advantage of expert systems is the separation of control reasoning (is this an attack?) from the formulation of the solution to the problem (system response to the attack). The disadvantage of expert systems is that they require a great deal of initial training and high maintenance during their lifetime. The initial rule-base must be generated by an expert which is time-intensive and expensive. Because not every expert knows every vulnerability in a system, there is the very real chance that the initial configuration does not capture all possible vulnerabilities. As new attacks are developed, the expert system must be manually updated to capture the characteristics of the new attacks.

Predictive pattern-based detection is an anomaly detection technique that attempts to predict future events based on events that have already occurred. Event sequences are represented as a statistically weighted set of rules based on a user profile. If user actions match n-1 events and the n^{th} event is statistically anomalous, then the system reports an intrusion. Predictive pattern systems constantly update user profiles and prune the rule set to maintain high quality patterns of user activity [10].

Neural networks are an anomaly detection technique that trains a neural network to predict a user's actions given a window of n previous actions. The network is trained through a user profile of representative user commands. If a user's actions are significantly deviant from the user profile as maintained by the neural network, the system reports an intrusion [11].

Machine learning is an anomaly detection technique that compares the user-input stream with a historical library of user commands to detect anomalous behavior. In one approach, the input stream is broken into fixed length sequences (normally 8-12 command tokens) which are compared through a sliding window against a library of 500-2000 user sequences. The library is unique for each user. The result of the comparison is a similarity measure. If the similarity measure is greater than threshold level, then the user activity is characterized as abnormal; otherwise, user activity is classified as normal [12].

The selection of several parameters greatly influences the effectiveness of a machine learning system. The optimal sequence length appears to be 8-12 command tokens. Shorter sequences provide low detection rates while longer sequences increase the false positive rate and provide lower intrusion detection rates. The sliding window size determines the shortest interval in which the system can detect an intruder. Experimental results also suggest that: the ideal library size is user dependent; as the size of the library increases, the number of false positives also increases; and, the method of pruning the library significantly impacts on the effectiveness of the overall system [12].

c) Graph-based Techniques: Graph-based techniques are misuse systems that represent user and system behavior as a set of graphs that are then compared to attack signature graphs to detect intrusions. This is a relatively new intrusion detection technique and has been used in a limited number of IDSs. There are three graph-based techniques used for intrusion detection: state transition analysis, pattern matching, and model-based detection.

State transition detection is a misuse detection technique that models a host as a state transition diagram. It was used as the basis for the USTAT system [13]. Known attack patterns are encoded as states in the diagram with the final state in a chain being the *compromised* state. The preceding states are known as *guard* states. The guard states act as a filter to separate normal from intrusive activities [13].

Pattern matching detection is a misuse system that represents known attack signatures as patterns that are compared against audit records. Knowledge about attacks is represented as a set of specialized graphs. The graphs represent the transition from normal system states to compromised states and are an adaptation of colored Petri nets. This technique is similar to the state transition technique, but pattern matching associates guards with transitions, rather than with states. This technique has been implemented in the Intrusion Detection In Our Time

(IDIOT) system in which pattern matching is used as the basis for a generic misuse detection model [9, 14].

Model-based detection is a misuse detection technique that detects attacks through observable activities that infer an attack signature. Model-based detection has three components: an *Anticipator*, *Planner*, and *Interpreter*. The *Anticipator* uses two types of models, activity models and scenario models, to predict the next expected step in an attack scenario. Activity models are representations of current activity while scenario models represent intrusion signature specifications. The *Planner* takes the *Anticipator's* prediction as a hypothesis and translates it into audit log format. These predicted audit entries are then used by the *Interpreter* as search strings in the audit records. If the model-based detection system accumulates sufficient evidence of an intrusion by crossing a system-defined threshold, the system reports an intrusion attempt [15].

d) Information Retrieval Techniques: Information retrieval, as used in intrusion detection, is a misuse detection technique that searches for attack patterns by building an index of audit logs and then searching this index. The information retrieval system must maintain the audit index by periodically rebuilding the index as new audit records are generated. There are a variety of techniques for building, searching, and storing indexes that result in different tradeoffs in terms of false positives and negatives [16].

e) Positive Behavior-Based Detection: Positive behavior-based intrusion detection is a specification-based technique that specifies intended system behavior and reports activity outside of this intended behavior [7]. This is one of the newest approaches to intrusion detection. There are two forms of positive behavior-based systems used for intrusion detection: specification-based and transaction-based detection.

Specification-based detection uses a program behavior grammar to enunciate intended behavior and then scans audit files for violations of this expected behavior. For example, the finger daemon in Unix should only execute the finger program and should only read a very limited subset of files that can be easily specified. If the finger daemon attempts to read the system password file, this violates program specification and an intrusion would be reported [17].

Transaction-based detection is a specification detection technique that delineates allowed actions and sequences of actions through transaction management. User activity is modeled as a series of read and write operations. The transaction-based detection system checks to ensure that all transactions are:

- Atomic (all operations are completed).
- Consistent (system remains in a consistent state).
- Isolated (transactions do not interfere with other transactions).
- Durable (transaction results are saved in permanent storage) [18].

As with the intrusion detection approaches, there is no one technique that provides complete security. As such, most modern IDSs employ two or more techniques to detect intrusions.

7. Host and Network-based Intrusion Detection

In executing the approaches and techniques discussed above, the IDSs can be either host-based, network-based or a combination of both approaches. Host-based IDSs monitor activity on a single computer using the host computer's audit information for analysis and detection. Network-based IDSs monitor network traffic to detect intrusions. Network-based IDSs are significantly more difficult to implement. From a bit stream representing network traffic, they must reconstruct connection, session, and application level traffic for all of the hosts on the network and detect intrusions in real time.

Both host-based and network-based IDSs suffer from a number of advantages and disadvantages. Host-based IDSs are typically easier to implement than network-based IDSs. However, host-based IDSs consume system resources that could be used for other user activities. Network-based systems do not consume user computing power but instead limit the impact of the IDSs to network bandwidth and the allocation of dedicated intrusion detection machines. Host-based detection is more readily subverted as it is an active agent that can be detected and attacked. Detection systems are prime targets for attackers. Network-based IDSs are more secure as they collect information passively and are more difficult for attackers to detect and defeat. Finally, host-based systems have limited visibility over intrusions that involve multiple hosts. This is a significant shortcoming as a number of common attacks are based on limited attacks on

Table 2: Classification of Existing Intrusion Response Systems

Intrusion Response Classification	# of Systems
Notification	30
Manual Response	8
Automatic Stateless Response	18
Total	56

multiple hosts. Network-based IDSs can detect multiple host intrusion attempts due to their greater visibility. Because of the limitations of both approaches, most IDSs use both host-based and network-based detection systems to provide more robust intrusion detection. No IDSs detects all intrusions and as such, IRSs must temper intrusion response generated with their confidence in the IDSs.

C. Intrusion Response Systems

1. Introduction

In the past seventeen years, a number of intrusion detection and intrusion response tools have been developed (See Appendix 1). The response systems can be categorized as notification systems, manual response systems, or automatic response systems (See Table 2). The majority of intrusion detection and response systems are notification systems only - systems that generate reports and alarms only. Periodic reports were the earliest form of intrusion response. Ranging in frequency from daily to monthly, reports record suspicious users so that the system administrator could further investigate potential intrusions. The frequency of reporting delimits the window of opportunity for an attacker. In today's environment, this window of opportunity is too large. Reporting, while still an important component of any intrusion response system, is not a viable means of intrusion response by itself. Alarms generate immediate messages to alert the system administrator to potential intrusive behavior. Alarms can be presented in a variety of formats including email messages, console alerts, and/or pager activations. After notification, further intrusion response is left as the responsibility of the system administrator.

Some systems provide the additional capability for the system administrator to initiate a manual response from a preprogrammed set of responses. While this capability is more useful than notification only, there is still a time gap between when the intrusion is detected and when the system administrator initiates a response. This window of exploitive opportunity is still too large.

Automatic response systems immediately respond to an intrusion and it is these systems that are most germane to this research. A survey of research literature found eighteen systems with automated response mechanisms. Fourteen of the systems with automatic response capabilities associate a specific response with a specific attack with no other formal reasoning capability other than the association. Four systems, Cooperating Security Managers (CSM), Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD), JiNAO, and Network Statistical Analysis Tool (NetSTAT) provide more robust intrusion response mechanisms through the use of decision components.

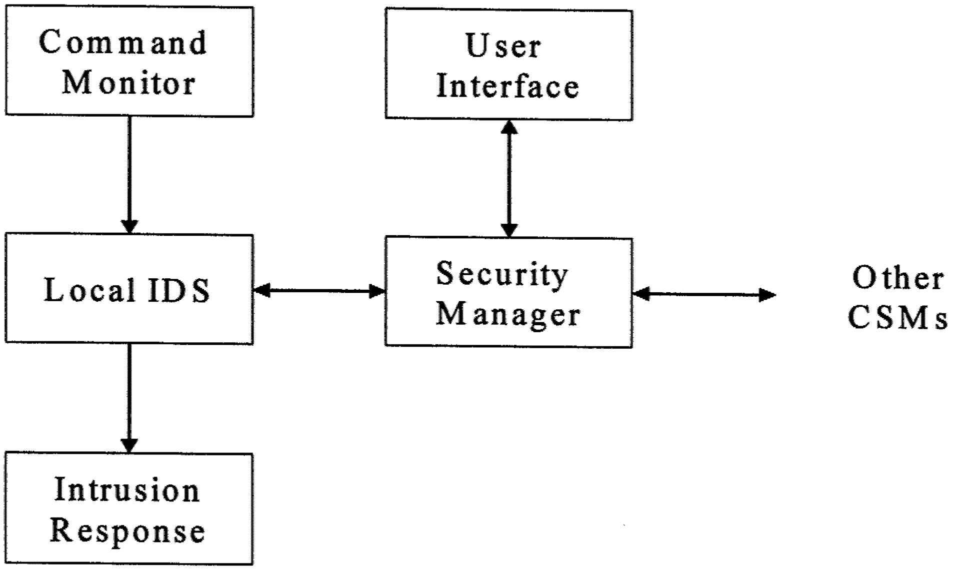


Figure 2: CSM Architecture

2. CSM

Cooperating Security Managers (CSM) is a distributed and host-based intrusion detection and response system (See Figure 2). CSM architecture consists of five components:

- Command Monitor: captures users commands and sends them to the Local IDS.
- Local IDS: a host-based detection system that looks for intrusions on the local system.
- Security Manager: examines network-related commands and coordinates with other CSMs to track connections and user activity.
- User Interface: provides the capability for the system administrator to query the Security Manager on the current security status.

- Intruder Handler: responds to detected intrusive behavior.

For every user and for the overall system, CSM maintains a suspicion level which indicates CSM's belief that a user is performing intrusive activity. If an intrusion is detected by the Local IDS or Security Manager, the Intruder Handler reacts to the intrusion by taking a preprogrammed action. At a minimum, the system administrator is notified. Depending on the intrusion, the intrusive session may perform a number of actions including terminating the current session or locking the user's account.

CSM provides automated responses through the Intruder Handler which consists of three different components: the Command Auditor, the Damage Control Processor (DCP), and the Damage Assessment Processor (DAP). The Command Auditor examines user command streams and automatically discards commands that it identifies as an attack. The DCP reactively responds to intrusive behavior using: (1) the Fisch Damage Control & Assessment (DC&A) taxonomy (See figure on page 41) to classify the attack; and (2), the suspicion level assigned to a user by the intrusion detection system. As the suspicion level changes, the DCP employs eight different response sets, each of which consists of one or more of fourteen different response actions. DCP continues to respond to intruder actions until the intruder leaves the system when their suspicion level is reset to zero [19, 20].

After the intruder leaves, the DAP attempts to restore the system to its pre-attack state and performs an attack analysis. Like the DCP, the DAP has eight response sets that it uses to restore the system. These response sets are associated with suspicion levels generated by the DCP. System restoration includes such actions as replacing modified

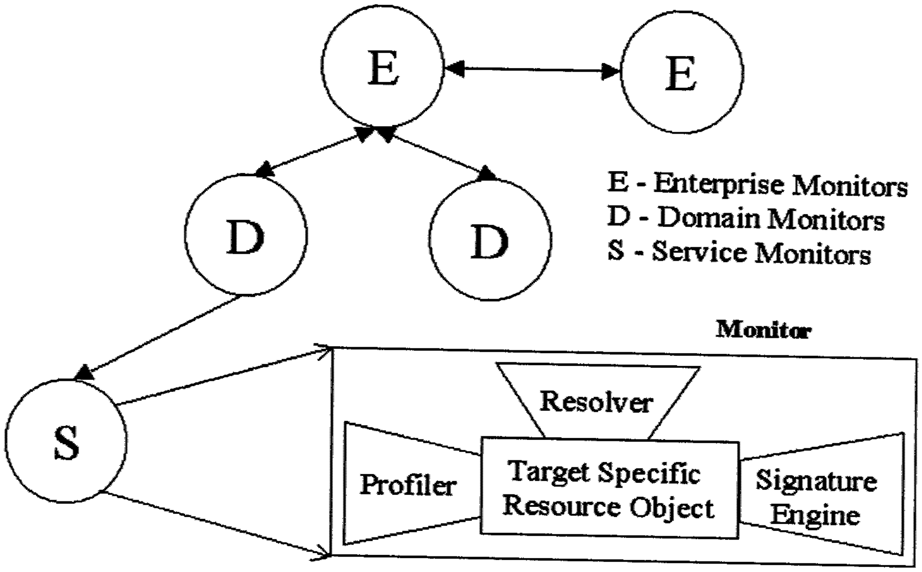


Figure 3: EMERALD Architecture [21]

files, removing new files, reconstructing system settings, and securing vulnerable user accounts [19, 20].

3. EMERALD

The Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) is a distributed misuse and anomaly intrusion detection system (See Figure 3). The Emerald architecture consists of hierarchical collection of enterprise, domain, and service monitors and is intended for large-scale heterogeneous computing environments. There are four principal components in the each monitor:

- Profiler Engine: a statistical anomaly detection component.
- Signature Engine: a signature -based inference component.

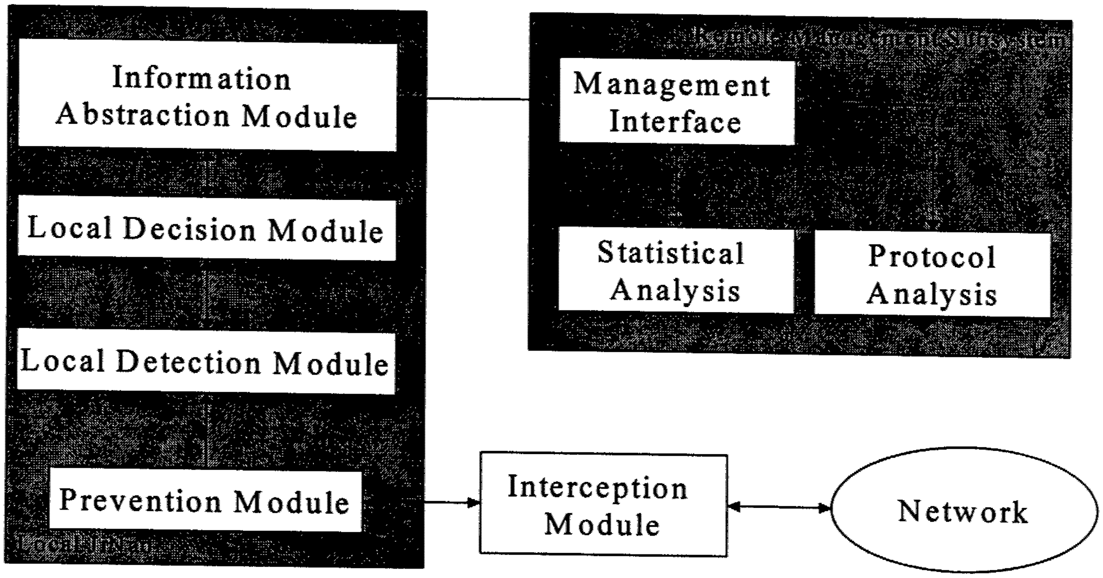


Figure 4: Ji-Nao Architecture

- Resource Object: a pluggable configurable library with all of the data for the other three components.
- Resolver: a coordinator of analysis and response policy enforcer.

Intrusion response is provided through the resolver. The resolver is an expert system that receives reports from the analysis components and invokes various response handlers. The possible responses are defined in the resource object with two associated metrics that delimit their usage: a threshold metric and a severity metric. The threshold metric defines the degree of intrusive evidence necessary to use the response. The severity metric defines how harsh a particular response is. Because of the system architecture, every monitor has an intrusion response capability [21, 22].

4. JiNao

JiNao is a misuse and anomaly-based intrusion detection system that attempts to detect attacks against the network infrastructure (See Figure 4). The JiNao architecture consists of two principal components: the Local Subsystem (LS) and Remote Management Subsystem (RMS). Each LS protects a single router or switch and consists of five components:

- Interception/Redirection Module: a routing component that redirects target protocol information flows to the prevention module.
- Prevention Module: a small-rule-based system that filters packets with clear security violations before the packets are processed by a router or switch.
- Local Detection Module: an expert system to detect attack signatures as well as a statistical analysis subsystem to detect anomalous behavior.
- Local Decision Module (LDM): the coordinator of the prevention module and local detection modules as well as the automated intrusion response component.
- Information Abstraction Module: interface component for the LS to other LSs and RMSs.

Each RMS consists of three components: a statistical analysis module, a protocol analysis module, and a management interface. RMSs coordinate the activities of several LSs and provide a set of management applications for the system administrator.

JiNao also supports intrusion response through reports, alarms, and automated response. Reports and alarms can be delivered through email or the JiNao GUI. The LDM may take automated defensive measures if an intrusion is suspected or detected by

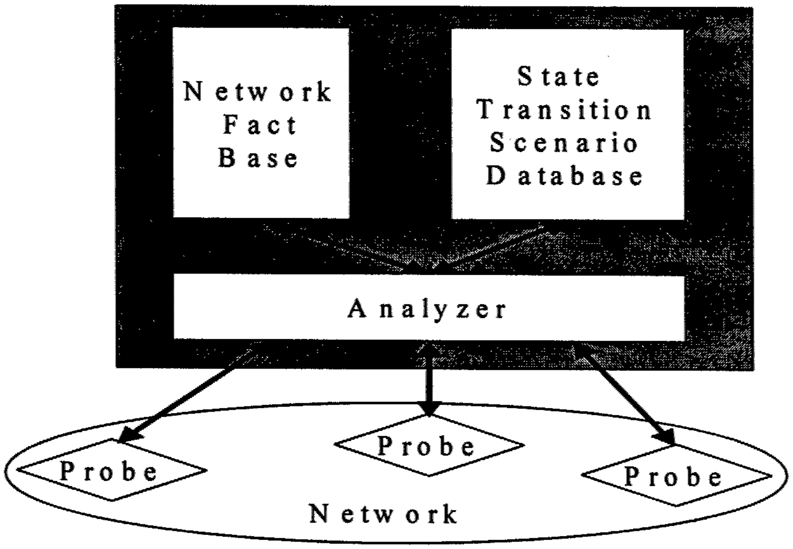


Figure 5: NetSTAT Architecture

increasing logging, disabling router interfaces, or undoing the effects of recent route update changes [23].

5. NetSTAT

Network Statistical Analysis Tool (NetSTAT) is a network-based misuse intrusion detection system (See Figure 5). NetSTAT represents attack signatures as state transition diagrams and extends previous research in the use of state transition analysis from host-based intrusion detection (see USTAT [13, 24]) to support network-based detection. Probes capture network traffic and compare activity against pre-programmed attack signatures. If an attack is detected, each probe has a local decision engine which is responsible for initiating intrusion response. This response can be in the form of reports, alarms, suggestions for system administrator action, or an automatic response by injecting datagrams into the network [25, 26].

D. Security Taxonomies

A taxonomy is a system with associated rules for classification of events into categories [27]. There has been research into a number of proposed security flaw taxonomies including the Research in Secured Operating Systems (RISOS) security taxonomy, the Protection Analysis (PA) taxonomy, Landwehr's taxonomies, Aslam's taxonomy, Bishop's taxonomy, and the Lindquvist taxonomy. These taxonomies classify security flaws which are an important component of an intrusion response taxonomy and as such are discussed below. An intrusion response taxonomy is the categorization of possible offensive and defensive responses to an intrusion. There has been only one published intrusion response taxonomy - the Fisch DC&A taxonomy. The DC&A

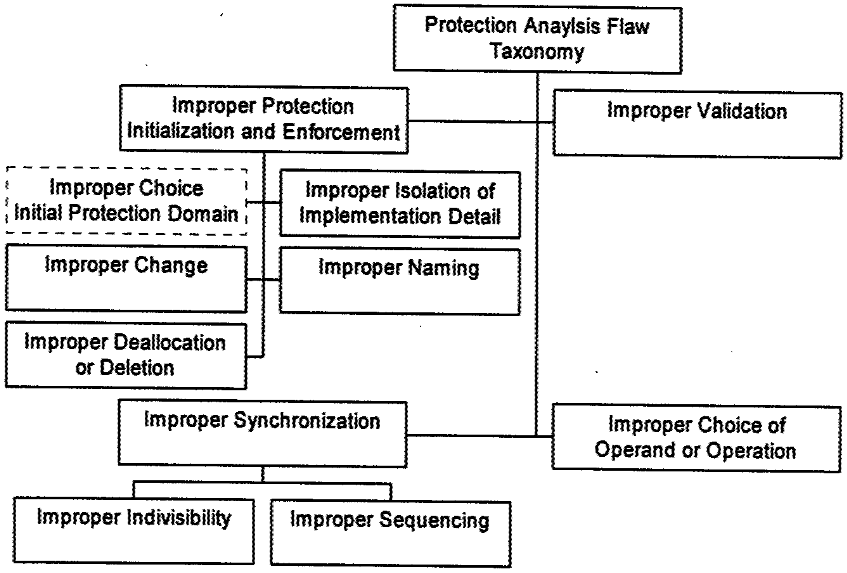

Protection Anaylsis Flaw Taxonomy
Improper Protection Initialization and Enforcement
Improper Validation
Improper Choice Initial Protection Domain
Improper Isolation of Implementation Detail
Improper Change
Improper Naming
Improper Deallocation or Deletion
Improper Synchronization
Improper Choice of Operand or Operation
Improper Indivisibility
Improper Sequencing

Figure 6: Protection Analysis Taxonomy

taxonomy is likewise addressed below.

1. Protection Analysis (PA) Taxonomy

The first published security taxonomy was the Protection Analysis (PA) taxonomy (See Figure 6). The objective of the PA project was to enable anyone to discover security errors by using an automated, pattern-matching approach. The taxonomy was based on the examination of over 100 flaws, found in six different operating systems, which the PA taxonomy categorized into ten categories. These ten categories were later reorganized into four different global categories:

- Improper Protection: This category includes flaws such as: incorrect installation of software; allowing users to bypass system controls and directly manipulate system data structures; "time-of-check-to-time-of-use" (TOCTTOU) flaws; allowing different objects to have the same name; and, leaving old data in deallocated memory.
- Improper Validation: This category encompasses buffer overflows and, those errors that involve not checking critical parameters and conditions that lead to system compromise. Buffer overflow attacks are attacks that attempt to overflow fixed sized data structures in programs. If these data structures overflow, the program will perform in an unexpected manner that may lead to system compromise.
- Improper Synchronization: This category addressed flaws that allow interruption of atomic operations or allow actions in an incorrect order.

- Improper Choice of Operand or Operation: This category includes an application's use of inappropriate operands or operations that leads to system's compromise.

While this taxonomy provided an initial classification of security flaws, the categories were too broad to be used effectively in an automated system and the same flaw could be classified in multiple categories. The pattern-matching approach used resisted automation and the underlying security fault database was never published [28-30]. The contribution of this study was the introduction of several types of security flaws that remain relevant. TOCTTOU, allocation/deallocation of residuals, and serialization errors were introduced and the group's research was an important step in the classification of security flaws [31].

2. RISOS Taxonomy

The Research in Secured Operating Systems (RISOS) security taxonomy categorized operating system flaws found in three operating systems (IBM's OS/MVT, UNIVAC's 1100 Series Operating System, and the TENEX system for the PDP-1) into seven categories (see Figure 7):

- Incomplete Parameter Validation: Parameters must be validated for data type, number, order, value and range. Failing to check if an array index is within the range of the array is an example of this type of flaw and can lead to system compromise.

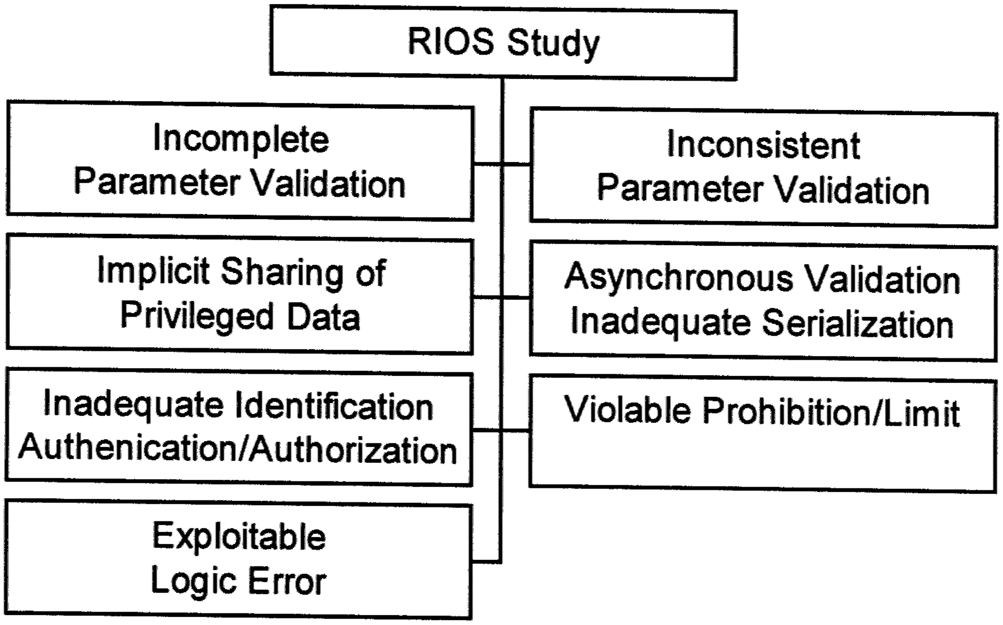

RIOS Study
Incomplete Parameter Validation
Inconsistent Parameter Validation
Implicit Sharing of Privileged Data
Asynchronous Validation Inadequate Serialization
Inadequate Identification Authenication/Authorization
Violable Prohibition/Limit
Exploitable Logic Error

Figure 7: RIOS Security Flaw Taxonomy

- Inconsistent Parameter Validation: Parameters may be evaluated at multiple locations within a program. If the evaluation criteria are not consistent, system compromise can occur.
- Implicit Sharing of Privileged Data: This category is the use of covert channels such as sending confidential information by modulating the load average of the system.
- Asynchronous Validation/Inadequate Serialization: This category encompasses TOCTTOU errors that introduce a small timing window of vulnerability that attackers could exploit to compromise the system.
- Inadequate Identification/Authentication/Authorization: If a privileged program does not require a process or individual to authenticate their

identity, attackers can exploit this implicit trust relationship to compromise the operating system.

- Violable Prohibition/Limit: This category addressed buffer overflow attacks.
- Exploitable Logic Error: This category captured errors not addressed by other categories such as exploitation of instruction side effects. [28, 32].

The final report suggested administrative actions that could prevent unauthorized access to a system and methods to prevent disclosure of information. The principal contribution of this study was a classification of integrity flaws found in operating systems [31]. The categories, however, remain too broad for use in an automated system.

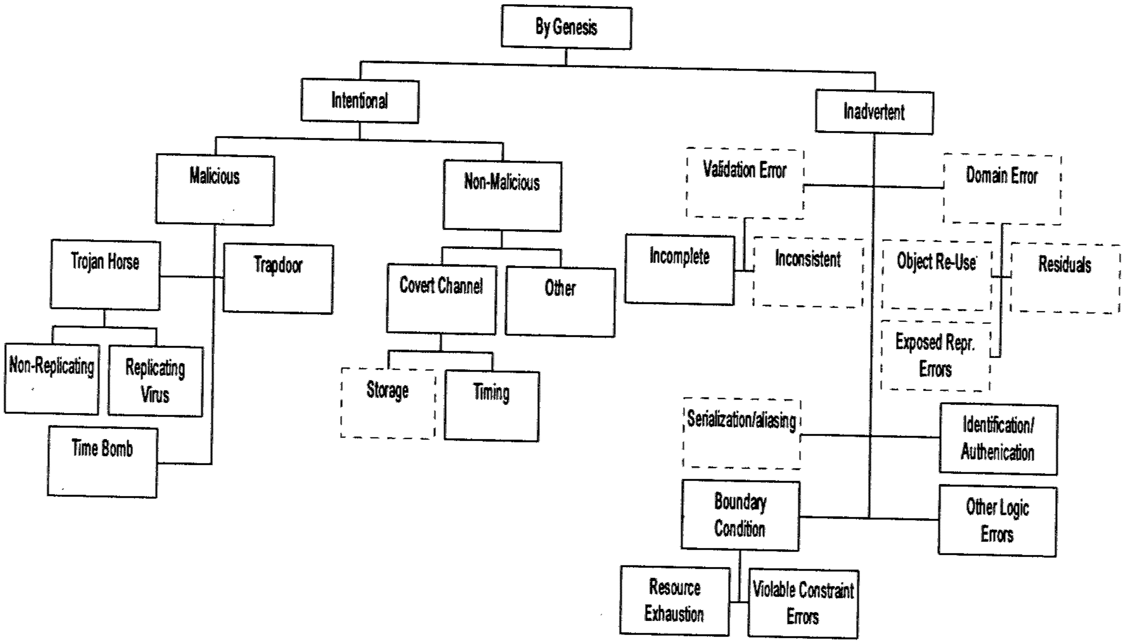


Figure 8: Landwehr Security Flaw Taxonomy (Flaw by Genesis)

3. Landwehr Taxonomies

Landwehr proposed three security flaw taxonomies categorizing flaws by genesis, time of introduction, and location (See Figures 8-10). The objective of these taxonomies was to describe how security flaws are introduced, when they are introduced, and where the security flaws can be found. This research also compared the frequency of security incidents against the taxonomies with the goal of helping software programmers and system administrators "to focus their efforts to remove and eventually prevent the introduction of security flaws" [30].

The Landwehr security flaw taxonomy by genesis extended the previous research of the PA and RISOS groups with the introduction of a new category of flaws, intentional flaws. Intentional flaws are flaws that are introduced deliberately into a program so that they can be exploited at a later time. Trapdoors, Trojan horses, time bombs, and covert channels are examples of intentional flaws. Inadvertent flaws in the Landwehr taxonomy were similar to the flaw taxonomies found in the PA and RISOS projects. The research found that validation errors were the most common security flaw followed closely by Trojan horses and domain errors [30].

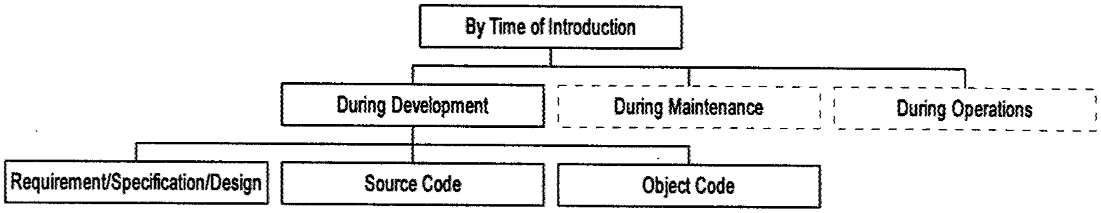


Figure 9: Landwehr Security Flaw Taxonomy (Flaw by Time of Introduction)

The Landwehr security flaw taxonomy by time of introduction characterized security flaws by when they were introduced into a system (See Figure 9). The researchers choose an abstract software development life cycle (SDLC) model to which a variety of SDLC models could be mapped. While other research characterized when in the SDLC that software defects were introduced, the Landwehr study was the first to describe when security flaws were introduced. The research found that most security flaws were introduced during development, followed by flaws during operations. The least number of security flaws were introduced during system maintenance.

The Landwehr security flaw taxonomy by location characterized security flaws by where the security flaw occurred (See Figure 10). This taxonomy differentiated security flaws as either hardware or software and subdivided the software category into operating system, support, and application flaws. The research found that most security flaws involve process management or privileged utilities.

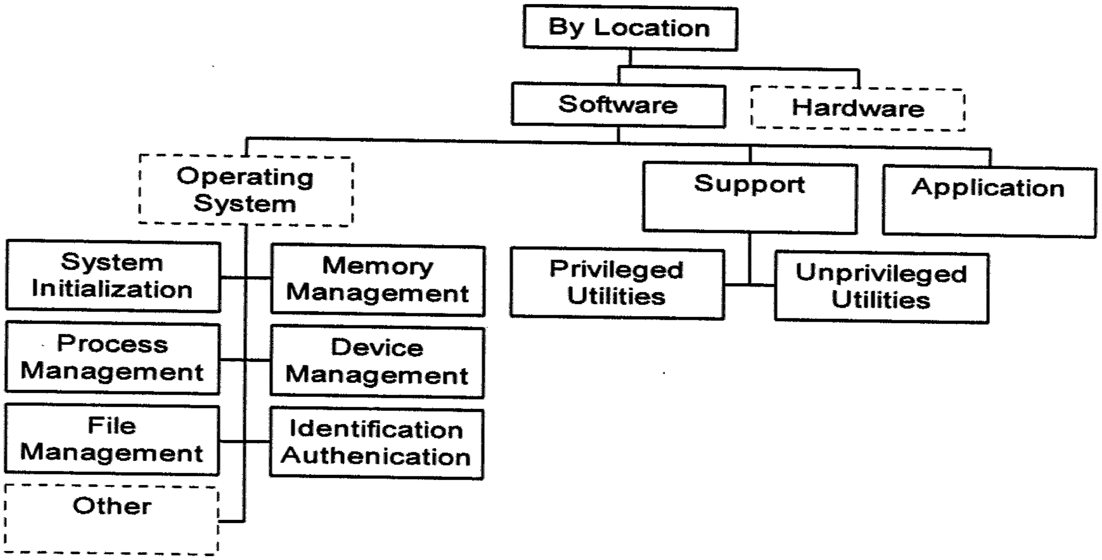


Figure 10: Landwehr Security Flaw Taxonomy (Flaw by Location)

The Landwehr taxonomies extended security taxonomy research by providing multiple taxonomies for characterizing security flaws. The realization that security flaws cannot be simply described by a single attribute was an important contribution and one that most future researchers have followed. Of the three taxonomies, only the by genesis taxonomy is germane to an automatic response tool. However, the genesis taxonomy requires a determination of intent to classify a flaw which is very difficult for an automated program to determine and virtually impossible in a real-time environment.

4. Bishop Taxonomy

Bishop studied flaws in the UNIX operating system and proposed a flaw taxonomy for the UNIX operating system. Rather than describe security flaws using a single set of categories, Bishop proposed that security flaws should be described using a single taxonomy that is composed of several collections of categories or axes. The proposed axes were:

- Nature of the Flaw: Bishop used the PA taxonomy for this axis.
- Time of Introduction: Bishop used the Landwehr security flaw taxonomy by time of introduction but modified Landwehr's definitions of the categories to more specifically define "during development", "during operations", and "during maintenance".
- Exploitation Domain: This axis measures the difficulty of exploiting a flaw by characterizing whether the flaw can be exploited using a program, a high-level user command language, or a configuration file.

- Effect Domain: The amount of access the attacker requires to implement the attack. Bishop divided this axis into four categories: (1) no special access; (2) network session; (3) physical (hardware) access; and (4) network sessions and physical access.
- Minimum Number of Components: The minimum number of components to exploit the vulnerability is the fifth axis in Bishop's taxonomy and indirectly measures the difficulty of detecting an attack by measuring the number of audit records that must be checked to determine that the attack took place.
- Source of the Identification of the Vulnerability: Bishop's last category identified where the flaw was first published. Bishop reasoned that it is important for misuse database compilers to know where to look to find the initial source of information on a flaw [28].

Bishop extended security flaw taxonomy research by including a number of criteria that previously had not been considered.

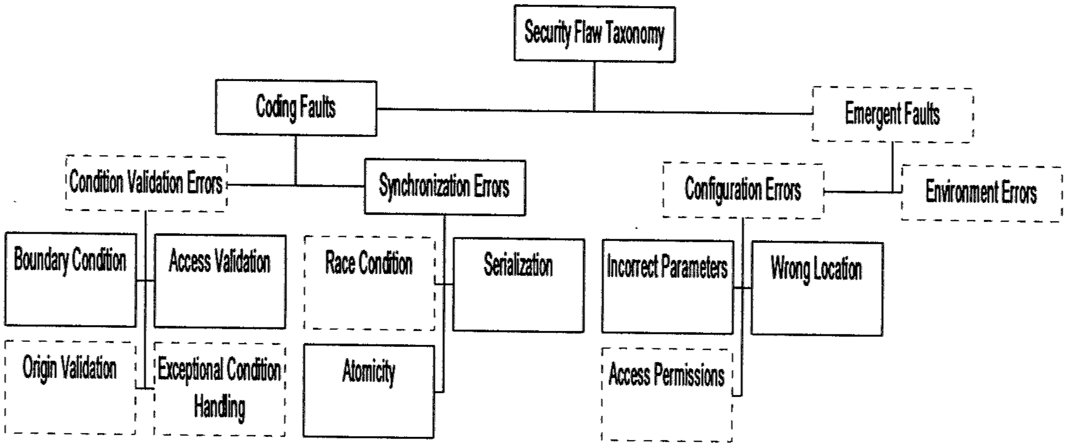


Figure 11: Aslam Security Flaw Taxonomy

5. Aslam Taxonomy

Aslam proposed a security taxonomy to classify the faults found in the UNIX operating system (See Figure 11). The objective of this taxonomy was to unambiguously classify security faults and provide a theoretical basis for the data organization of a vulnerability database. Selection criteria are provided for each criterion so that all fault categories are specific and distinct. The Aslam taxonomy contained the following major categories:

- Coding Faults: Coding faults are flaws that are introduced during software development.
  - Condition Validation Errors: A flaw is a conditional validation error if the fault results from a missing or incorrect check for limits, check for access rights, check for valid input, or authentication check.
  - Synchronization Errors: A flaw is a synchronization error if the fault results from improper serialization of operations or the existence of a timing window between two operations that can be exploited.
- Emergent Faults: Flaws that result from improper installation of software, unexpected integration incompatibilities, and when a programmer fails to completely understand the limitations of the run-time modules.
  - Configuration Errors: A configuration error occurs if a program is installed in the wrong location, installed with incorrect setup parameters, or installed with incorrect permissions.

- Environmental Errors: Environmental errors occur when modules perform according to specification but an error occurs when they are subjected to a specific set of inputs in a particular configuration environment.

The Aslam taxonomy was used as the theoretical basis for a vulnerability database that was used in the Intrusion Detection In Our Time (IDIOT) IDS [31, 33, 34].

6. Lindqvist Taxonomy

Lindqvist and Jonsson proposed two taxonomies that differed from previous work in that they characterized security attacks based on the technique used and the

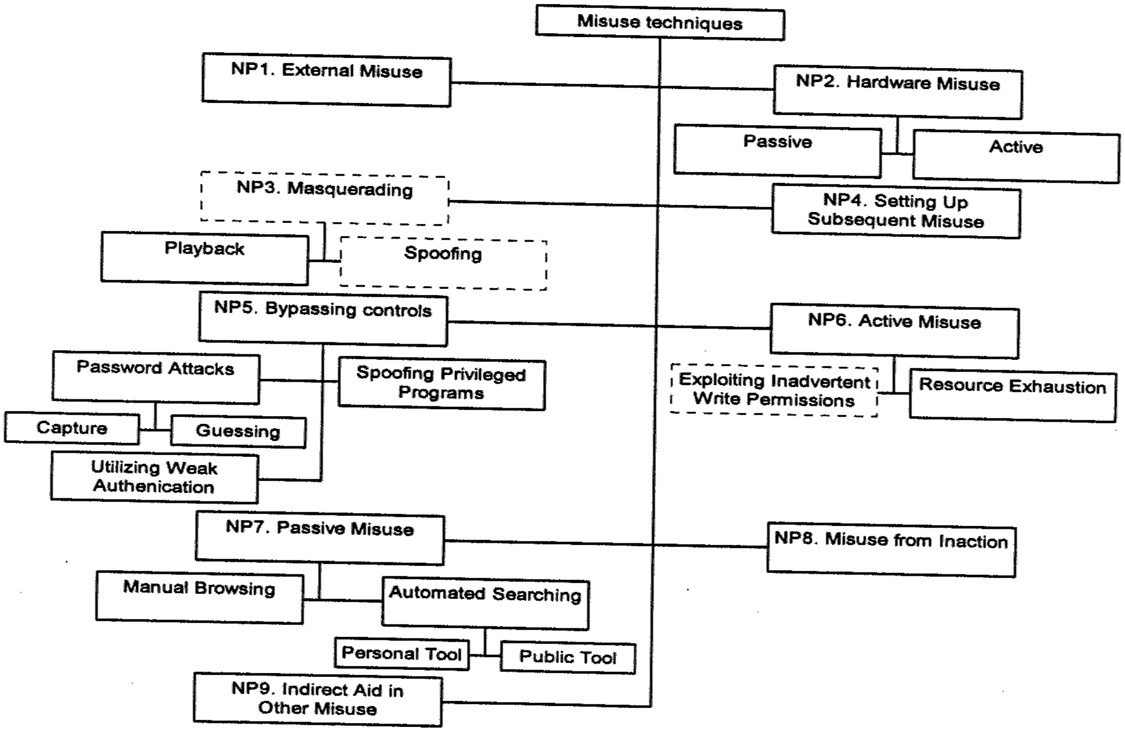


Figure 12: Lindqvist Intrusion Technique Taxonomy

result of the attack. The objectives of Lindqvist and Jonsson research were threefold: (1) to establish a framework for the systematic study of computer attacks; (2) to establish a structure for reporting computer incidents to an incident response team; and, (3) to provide a mechanism for assessing the severity of an attack.

The Lindqvist Intrusion Technique Taxonomy is based on previous research by Neumann and Parker [35] and divided intrusive techniques into three principal categories (See Figure 12):

- Bypassing Intended Controls: This category includes attempts to attack passwords, spoof privileged programs, and attack programs utilizing weak authentication.
- Active Misuse of Resources: This category includes active attacks such as buffer overflows as well as exploitation of world writeable system objects.

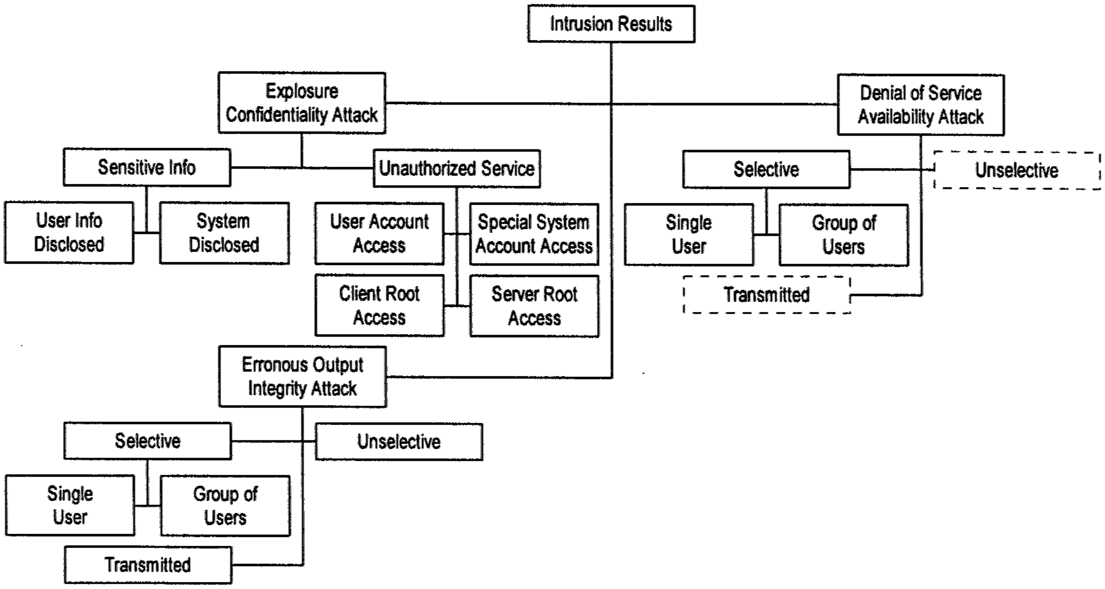


Figure 13: Lindqvist Intrusion Result Taxonomy

- Passive Misuse of Resources: This category includes all probing attacks that attempt to identify weaknesses in the scanned system [36].

The Lindqvist Intrusion Result Taxonomy is based on the Confidentially, Integrity, and Availability (CIA) model [36]. It divided intrusion results into three categories (see Figure 13):

- Exposure: These are attacks against system confidentially and are subdivided into disclosure of confidential information and service to unauthorized entities.
- Denial of Service: These are attacks against system availability and are subdivided into selective, unselective, and transmitted attacks. Transmitted attacks are attacks that affect the service delivered by other systems to their users.
- Erroneous Output: These are attacks against system integrity and are subdivided into selective, unselective, and transmitted attacks [36].

The Lindqvist Intrusion Result Taxonomy use of the widely respected CIA model provides a good theoretical foundation for the classification of intrusion results. Intrusion results are an important component of an automatic intrusion response system as the response should be tailored to the attack. As such, the Lindqvist Intrusion Result Taxonomy will be included as a component of the AAIR intrusion response taxonomy.

7. Fisch DC&A Taxonomy

A review of literature reveals only one intrusion response taxonomy - the Fisch DC&A taxonomy. The Fisch DC&A taxonomy classified the intrusion response according to: when the intrusion was detected (during the attack or after the attack); and, the response goal (active damage control, passive damage control, damage assessment, or damage recovery) [19]. This taxonomy only provided for defensive intrusion responses and did not categorize offensive responses. It also did not consider the type of attack, type of intruder, sensitivity of the information being attacked, or environmental constraints in formulating a response (See Figure 14). While the categories covered by the Fisch taxonomy should be components of any future intrusion response taxonomy, additional components are necessary to more accurately classify intrusion responses.

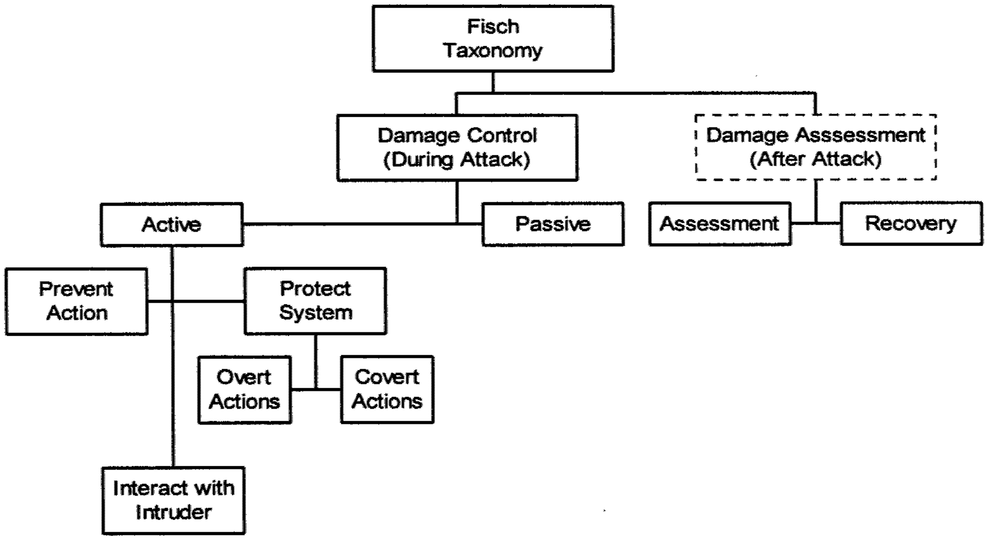


Figure 14: Fisch DC&A Intrusion Response Taxonomy

E. Software Agents

There have been a number of definitions of software agents in the past twenty-five years. Franklin and Graesser described agents through a set of properties rather than through a general definition (See Table 3). To be considered a software agent, programs must satisfy the first four properties and are characterized according to their properties [37]. For example, this research will employ non-mobile, communicative learning agents. Gilbert defined agents using three dimensions: agency, intelligence, and mobility. Agency is the degree of autonomy and authority vested in the agent and can be measured by the nature of interaction with other entities in the system. Intelligence is the degree of reasoning and learned behavior: the agent's ability to accept user's statement of goals and carry out the task delegated to it. User model is an indication as well as ability to learn and adapt. Mobility is the degree to which agents themselves travel through the network [38].

Table 3: Characteristics of Software Agents [18]

Property	Explanation
Reactive	Responds in a timely fashion to changes in the environment
Autonomous	Exercises control over its own actions
Goal-Oriented	Does not simply act in response to the environment
Temporally Continuous	Is a continuously running process
Communicative	Communicates with other agents, perhaps including people
Learning	Changes its behavior based on its previous experience
Mobile	Able to transport itself from one machine to another
Flexible	Actions are not scripted
Character	Believable "personality" and emotional state.

Russell and Norvig proposed four categories of intelligent agents: reflexive, reflexive with internal state memory, goal-directed, and utility-based. Reflexive agents make a predefined, immediate response based on the current environmental state. Reflexive agents with internal memory supplement environmental state with a memory of previous actions. Goal-directed agents are agents with multiple and often competing goals while utility-based agents attempt to maximize a utility function by choosing the alternative with the highest utility value [39]. With the exception of CSM, all current intrusion response systems can be classified as reflexive systems or utility-based systems. There are no intrusion response systems that incorporate goal-directed agents or a combination of the previously mentioned agent categories. This research addresses these issues by providing a methodology for reflexive agents with internal memory, utility-based, and goal-directed agents to cooperatively respond to intrusive behavior.

Franklin and Graesser, Gilbert, and Russell and Norvig provide just three definitions of agency. Maes [40-42], Coen [43], Nwana [44], and others have generated different definitions. For purposes of this research, the characterization approach of Russell and Novig will be used.

F. Agent Communication

There are two main approaches to designing an agent communication language [45]. One approach is based on executable content using programming languages such as Java or Tcl. Agents communicate with their own procedural language that is understandable by the other agents in the system. When the amount of information that is transmitted between agents is relatively small, this approach works well. The second approach is more declarative and uses popular agent languages such as KQML. This approach works nicely when large amounts of information need to be transmitted between agents in a standardized way.

CHAPTER III

DESIGN

A. Introduction

The design of an adaptive intrusion response system consists of an intrusion response taxonomy and an associated methodology. The taxonomy provides a theoretical classification of responses necessary for an automated system. The methodology describes the conceptual model of the adaptive intrusion response system. The intrusion response taxonomy and components of the methodology are discussed below.

B. Intrusion Response Taxonomy

There are a number of responses to an intrusion that range from monitoring the intrusion to actively attacking the intruder. Not all responses are appropriate for all intrusions. For example, terminating the attacker's session after the attacker has already logged out will have no effect. As such, there is a need to categorize responses so that they are appropriate to the attack. A taxonomy is also necessary as it provides the necessary framework for automatic intrusion response. Landwehr et al. observed [30]:

> A taxonomy is not simply a neutral structure for categorizing specimens. It implicitly embodies a theory of the universe from which those specimens are drawn. It defines what data are to be recorded and how like and unlike specimens are to be distinguished.

The taxonomy is composed of a number of dimensions where each dimension provides a categorization necessary for formulating an appropriate response. There are six dimensions: response timing, type of attack, type of attacker, strength of suspicion, implications of the attack, and environmental constraints (See Figure 15).

1. Response Timing

The timing of the response is a fundamental delineation in formulating a correct response and as such, it is the first dimension of the taxonomy. The response timing may be defined as preemptive, during an attack (damage control), or after an attack (damage assessment). Preemptive responses occur when there are indications of an attack but the attack has not actually begun. Preemptive responses attempt to increase the defensive posture of the potentially affected system while continuing to provide service to users with minimal degradation of performance. Damage control responses occur when the

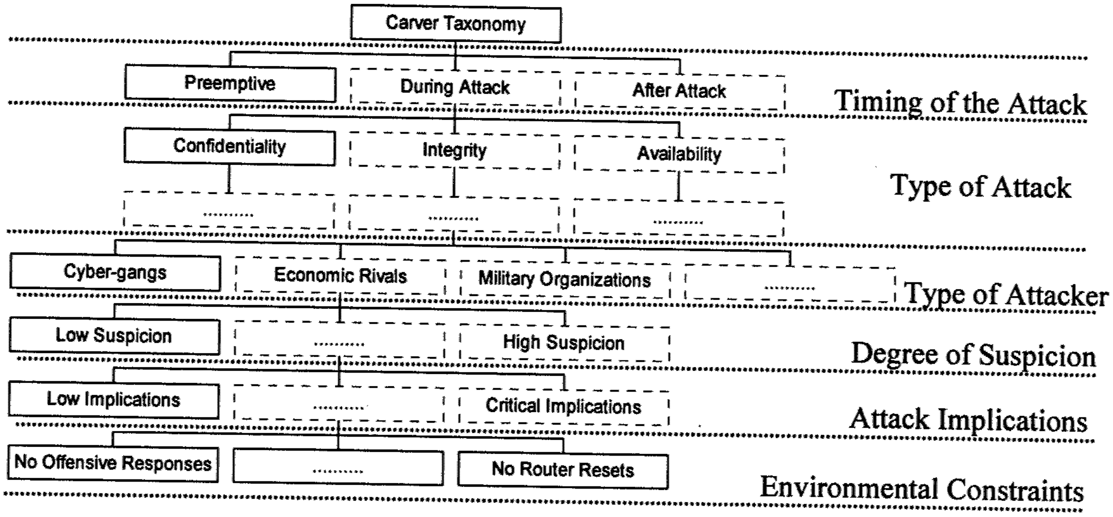


Figure 15: Partial Representation of Carver Intrusion Response Taxonomy

attack has been detected and is ongoing. These responses attempt to limit the effect of the attacker while continuing to provide service to legitimate users. Damage assessment responses occur when the attack was detected after the attacker has left the system. These responses attempt to document and repair any damage to the attacked system.

2. Type of Attack

The type of attack is an important characterization in determining an appropriate response. For example, the response to a denial of service attack is different from a race condition attack involving a system utility. There is no attack taxonomy that is both complete (encompasses all possible attacks) and correct (appropriately characterizes attacks). The best characterization of the type of attack is the Lindqvist Intrusion Result Taxonomy [36](see Figure 13). It uses the CIA model as a theoretical basis for determining the type of attack and provides the necessary differentiation between the types of attacks for automatic intrusion response. As such, it is used as the second dimension in the intrusion response taxonomy.

3. Type of Attacker

The type of attacker is similarly an essential characterization in determining an appropriate response to an attack. For example, there is a difference in responding to a novice attacker using a well-known attack script and a distributed, coordinated computer attack supported with the computational resources of a nation-state. Differentiation between type of attacker is a difficult task but fundamental to the formation of an appropriate response. The most useful distinctions are:

- Is the attacker a novice or expert attacker?
- Is the attacker using an automated program or is it a manual attack?

Certain techniques such as locking a user account or using remote logging will be very effective against a novice attacker while they will have almost no effect against an expert intruder. Similarly, automated attack programs can be easily disrupted by forcing additional authentication or similar techniques while they will have limited effect on a manual attack. As such, the classification of the type of user as a novice/expert and automated/manual attacker is included as the third dimension of the response taxonomy.

4. Strength of Suspicion

The fourth dimension of the intrusion response taxonomy is the strength of suspicion. Current intrusion detection is not an exact science and as a result, intrusion detection systems can generate false positive or false negative results. Some user activity is clearly intrusive while other activity may be indicative of intrusive behavior or may be normal user activity. The response must be tempered by the strength of suspicion that an actual intrusion is occurring. If the degree of suspicion is low, the response may be limited to account for the possibility of a false positive detection. If the degree of suspicion is high, a broader range of responses is possible. As such, the strength of suspicion is a key component of an intrusion response taxonomy.

5. Implications of the Attack

The fifth dimension of the intrusion response taxonomy is the implications of the attack. Different systems have differing degrees of importance within an organization.

This difference in criticality should lead to different responses, to the same attack, against different targets. For example, the response should be different if it is a denial of service attack against a single workstation as compared to the same attack against an institutional Domain Name Server.

6. Environmental Constraints

The final dimension in the intrusion response taxonomy is environmental constraints. There are legal, ethical, institutional, and resource constraints that limit what responses are appropriate. For example, current U.S. law prohibits launching a counterattack against a suspected attacker. This constraint does not apply during a declared war when the counterattack is part of a military operation. The environmental constraints are an important consideration in the formulation of a response and as such are included as a dimension in the intrusion response taxonomy.

C. Methodology

The methodology for adaptive intrusion response is summarized in Figure 16. Multiple IDSs monitor a computer system and generate intrusion alarms. *Interface agents* translate IDS detection messages into a common message format and maintain a model of each IDS based on number of false positives/negatives previously generated. It uses this model to generate an attack confidence metric and passes this metric along with the intrusion alarm to the *Master Analysis agent*. The *Master Analysis agent* classifies whether the incident is a continuation of an existing incident or is a new attack. If it is a

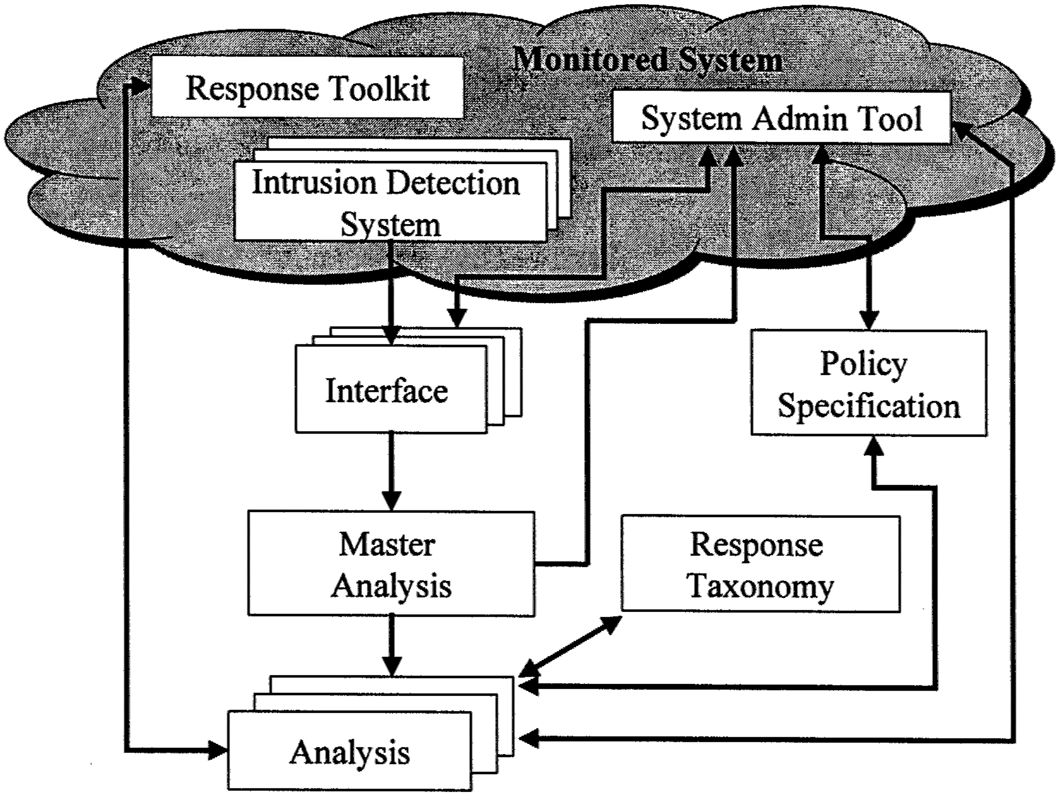


Figure 16: Methodology

new attack, the *Master Analysis agent* creates a new *Analysis agent* to develop a response plan to the new attack. If the incident is a continuation of an existing attack, the *Master Analysis agent* passes the attack confidence metric and intrusion alarm to the existing *Analysis agent* handling the attack. The *Analysis agent* analyzes an incident until it is resolved and generates a course of action to resolve the incident. To generate this course of action, the *Analysis agent* involves the *Response Taxonomy agent* to classify the attack and *Policy Specification agent* to limit the response based on legal, ethical, institutional, or resource constraints. The *Analysis agent* also decomposes the abstract course of action into very specific actions and then invokes the appropriate components of the *Response Toolkit.* The *Analysis agent* employs adaptive decision-making based on the success of previous responses. Each of the various components of the methodology is discussed below.

1. Intrusion Detection System(s)

One or more IDS(s) detect intrusions and generate intrusion reports. Because the focus of this research is on intrusion response and not intrusion detection, this component is simulated.

2. Interface Component

The interface module performs two functions: it translates IDS specific messages into a generic message format and it maintain a confidence metric on the reporting IDS. There are two techniques for message formats: a general communications language such as the Knowledge Query and Manipulation Language (KQML) or Common Intrusion Detection Format (CIDF); or, the use of a specialized language [38]. Because the architecture must interact with a variety of different IDSs, there is no assumption of a common communications language. The interface component provides this translation service so that all messages internal to the response system are in a common format. Additionally, due to the requirement for rapid intrusion response, a specialized language is used internal to the response system instead of a generalized language. In an intrusion response system, the efficiency and speed a specialized language provides is more important than the flexibility and interoperability that a generalized language provides.

The interface module also maintains a confidence metric on the reporting IDS. IDSs are not perfect and will generate false positive and false negative alarms. The response must be tailored by the degree to which the response system believes that the reported incident is a real attack and not a false alarm. The confidence metric is the ratio of false positive reports to actual reports. The number of false positives is generated through a feedback loop between the interface component and the system admin tool. After each incident, the system administrator can indicate whether the incident was a real attack or a false alarm. This results in an update to the confidence metric for the reporting IDS and over time, response adaptation. Responses to incidents from IDSs that generate a high number of false positives will be less severe than reports from IDSs that seldom generate false alarms. There is one interface module per IDS and the confidence metric generated by the interface module is passed to the Master Analysis component.

3. Master Analysis Component

The Master Analysis module examines the incident report generated by the interface component and determines whether the incident is a new attack or a continuation of an existing attack. If the incident is a new attack, the Master Analysis component creates a new analysis component and passes to it the incident report and associated confidence metric. If the incident is the continuation of a previously detected attack, the Master Analysis component simply forwards the incident report and associated confidence metric to the appropriate analysis component.

The classification of the incidents as part of an ongoing attack or a new attack requires reasoning under uncertainty. Some incidents such as multiple attacks from the same Internet Protocol (IP) address within a short interval of time provide a clear indication of the continuation of an existing attack. Attacks such as a distributed port scan from multiple IP addresses over several days or weeks would be much more difficult to detect. In determining if an attack is a continuation of the same attack or a new attack, the Master Analysis component uses three metrics: time, session identifier, and attack type. If the same attack is launched multiple times in a short period of time, it is reasonable to assume that the attacks are related. To a lesser extent, if the system has not been attacked for a period of time and suddenly is assailed by a number of different attacks in a short period of time, it can be inferred that the attacks are all part of the same incident. If the attack is from the same IP address or same subnet as a previous attack, this is a clear indication of a continuation of an attack. If the attack is from the same user there is a clear indication of the continuation of the same attack. Finally, if the same attack program or process is the source of the incident report, then it is likely that the attacks are related and part of the same ongoing attack.

It is not the intent of this research to impose a particular inference mechanism within the Master Analysis module but instead to advocate that there must be a classification of incidents. The prototype Master Analysis components constructed to validate this methodology used both crisp and fuzzy rule bases [46]. Fuzzy rule bases have the advantages that it is relatively easy to capture the knowledge of domain experts and later verify how the Master Analysis module reached classification decisions.

4. Analysis Component

The Analysis module provides long-term analysis of an incident and determines a plan to respond to an intrusion. This plan consists of a *response goal*, one or more *plan steps*, and associated *tactics* for accomplishing the plan steps. The *response goal* is specified by the system administrator and provides a general response approach. Examples of response goals include: catch the attack, analyze the attack, mask the attack from users, sustain service, maximize data integrity, maximize data confidentiality, or minimize cost. *Plan steps* are techniques for accomplishing a response goal. Examples of plan steps include: gather evidence, preserve evidence, communicate with the attacker, slow the attack, identify compromised files, notify the system administrator, or counterattack the attacking system. *Tactics* are methods to carry out a plan step. For example, given a plan step of gather evidence, there are a variety of tactics for accomplishing this plan step such as enabling additional logging, enabling remote logging, enabling logging to an unchangeable media, enabling process accounting, tracing the connection, communicating with the attacker, or enabling additional IDSs. The tactics can be further decomposed into a number of *implementations* that are environment dependent. As an example, consider a subnet consisting of the machines Limbo, Saint Peter, and Heaven. If Saint Peter is attacked, the tactic of remote logging could be implemented by logging to computer system Limbo or Heaven or both. The analysis agent determines what plan steps, tactics and implementations are appropriate. The analysis module makes this determination using several inputs:

- Confidence metric: The Analysis component receives the confidence metric from the Master Analysis component.
- Incident report: The Analysis component receives the incident report from the Master Analysis component and forwards it to the Response Taxonomy component. The Response Taxonomy component uses this information to weight various response options.
- Incident history: The Analysis component maintains a history of the incident and forwards this history to the Response Taxonomy component for proper classification. The type of attacker dimension, for example, depends on history of attacks attributed the attacker. The Analysis module maintains this information and provides to the Response Taxonomy component as needed.
- Response Goal: The system administrator sets the response goal of the system and the Analysis component uses goal to weight potential responses. Possible response goals are: analyze the attack, catch the attack, mask the attack, maximize confidentiality, maximize data integrity, minimize cost, recover gracefully, and sustain service.
- Plan history: The Analysis component maintains a history of previously implemented plans so that it does not implement a plan that had previously failed.
- Policy specification: The Analysis component coordinates with the Policy Specification component to ensure that the plan being pursued is in

compliance with the policy restrictions of the computing environment. These restrictions include legal, ethical, institutional, and resource-based constraints.

Given these inputs, the Analysis component develops a response plan. It then monitors the implementation of that plan and adjusts the plan if necessary. As the Analysis component receives additional incident reports, it has three options: continue with the same plan, adapt the plan, or replan completely. Continuation of the same plan is appropriate when there are indications that the plan is working or there is not enough evidence to support a change in plans. Adaptation of the plan is appropriate when there are significant changes in the environment or significant failures in the plan. Replan completely is appropriate when adaptation of the plan is not sufficient given the required changes. If all other measures have failed, the analysis module will shut down the host to protect the machine until the system administrator can actively diagnose the damage done to the machine.

Each plan step, tactic and implementation (PTI) has associated with it a success metric which is the ratio of successful responses to an intrusion to the total number of responses using a particular PTI. This metric is updated by the system administrator after each attack and the system dynamically adjusts what plans are selected to respond to an intrusion. Those PTI that are more successful are weighted so that they will be used more often than PTI that the system administrator determines were not successful.

Plan steps include:

- Gather Evidence: One of the first steps in any intrusion is to gather evidence so that the system administrator can identify affected systems and restore these systems to its pre-attack state.
- Preserve Evidence: Attackers often attempt to remove an indication of an attack from the system log files. The response system can thwart these attempts through a number of tactics such as logging to an unchangeable media or logging to a remote machine. This plan step is especially appropriate when trying to counter the attacks from expert intruders that have a high probability of removing any traces of their attack.
- Slow/Stop the Attack: Every response plan will attempt to either slow or stop the attack. Determining whether to slow or stop an attack depends on a number of factors that the analysis and response taxonomy module must consider such as the system goal, type of attacker, and type of attack. If the system goal were to analyze the attack of an expert attacker, then slow the attack would be preferred over stopping the attack. If the attacker is a novice using a simplistic attack and the system goal is to maintain service, then stopping the attack would be more appropriate.
- Identify Damaged Files: With some low priority systems, it is easier to restore the system after an attack then to try to actively defend the system. This restoration is easier if the affected files are identified.

- Protect Critical Files: There are techniques for limiting the damage to critical files when a system is under attack. Employing these techniques allows the system to limit potential damage or rapidly restore the system to its pre-attack state.
- Notify the System Administrator: Like the gather evidence plan step, notifying the system administrator is a high-priority response that is usually implemented. No matter how good the response system, a skilled system administrator is the best defense during an attack.
- Communicate with the Attacker/Employ Social Engineering on the Attacker: Attackers operate under the assumption that their attack has not been detected or that their identity is protected. These plan steps make it clear to the attacker that they have been detected and that the system is being defended. Social Engineering goes beyond normal communication with the attacker and instead attempts to manipulate the attacker so as to nullify or lessen the effect of the attack.
- Counterattack: If not constrained, often the best defense is an offense. If the attacker can be clearly identified, counterattacking makes the attacker defend his system and can divert the attention of the attacker.

Tactics to achieve the previously mentioned plan steps include:

- Generate a Report: All intrusive behavior should be logged so that it can be reviewed by a system administrator. These reports provide critical information for the resolution of ongoing incidents and facilitate long-term analysis of security attacks.

- Generate an Alarm: As previously discussed, the success of an attack is dependent on the time between detection and response. Alarms, implemented through email messages, console messages, pagers, or even loudspeaker announcements, notify the system administrator that an attack is underway. Not all intrusive behavior, however, should generate an alarm. SUDO is an authentication program that allows a normal user to perform a single command as root. There is a difference, for example, between a single failed SUDO attempt and one hundred failed SUDO attempts from the same user. The latter should generate an alarm while the former probably should not except on the most sensitive systems.
- Lock User Account: If a user account has been compromised, an appropriate response would be to lock that user's account so that it cannot be used to launch future attacks.
- Suspend User Jobs: If there are indications of intrusive behavior as well as normal user operations, the suspension of user jobs and termination of user sessions allows the system administrator the opportunity to terminate any intrusive jobs while not corrupting valid user tasks. While termination of user sessions without suspension of user jobs would be a more common response, there are circumstances when it would be desirable to suspend user jobs.

- Terminate User Session: If a user is involved in intrusive behavior, the user's session should be terminated and the user's account locked to prevent future damage.
- Enable Additional Logging: Some user behavior cannot be unambiguously characterized as intrusive behavior but is nonetheless indicative of possible intrusive behavior. In such cases, enabling additional logging allows for the gathering of additional information that may help in classifying the user's behavior.
- Enable Remote Logging: Additional logging may not be sufficient against certain types of attacks or attackers and instead, remotely logging to another system or a non-changeable media (such as CD-ROM or a printer) may be a better technique for gathering additional information on the attacker.
- Block IP Address: If the IP address of an attacking system can be identified, some network attacks can be neutralized by blocking, at a router, all traffic from that address. While this protection is often temporary if the attacker can change their IP address, it will slow the attacker and allow the intrusion response system or system administrator more time to respond to an attack.
- Enable additional intrusion detection tools: Because intrusion detection tools are imperfect and consume system resources, intrusion response systems may enable additional intrusion detection tools as the degree of

suspicion increases that an intrusion is ongoing. More robust and costly (in terms of resource utilization) detection tools can be employed (up to a point) as additional indicators of intrusive behavior are found.

- Shutdown Host: Sometimes the only mechanism for protecting against further system compromise is to shut down the machine. While this is a draconian measure, it is sometimes the only mechanism for protecting a host under an active attack.
- Disconnect from the Network: For network-based attacks, disconnecting from the network is less draconian than shutting down the host but has the same effect - network-based attacks can no longer effect the system allowing the system administrator time to respond to an attack and repair any damage to the attacked system.
- Disabling the Attacked Ports or Services: If a single service or well-known port is being used as the basis for the attack, that port or service can be disabled effectively stopping the attack without affecting any of the other services offered by the system.
- Warn the Intruder: Most attackers operate with the assumption that they are not being actively monitored or that they can evade intrusion detection systems. Telling the intruder that they are actively being monitored is all that is required for them to abandon the attack.
- Trace connection: Criminal prosecution of computer attackers, while a viable response to intrusions, is outside the scope of intrusion response

systems. However, tracing by the network connection of an attacker so that the attacker can be positively identified is a viable response. As a side effect, the attempt to trace back a connection can be detected by the attacker. For less experienced attackers, the fact that someone is actively trying to trace them will often result in the termination of the attack.

- Force Additional Authentication: Forcing additional authentication slows down or stops an attack while allowing authorized users to continue to use the affected system. The suspected intruder must provide additional proof of their identity before they can execute commands.
- Create Backups: Attacks against the integrity of a system can be thwarted by creating up-to-date system backups for system restoration and file comparison. While it is often impractical to maintain real-time backups of all modified files, as the degree of suspicion that the system is being attacked increases, the time interval between backups should be decreased so as to limit lost or corrupted data.
- Employ Temporary Shadow Files: A temporary shadow file is a duplicate file created and encrypted to protect the original file. When an intruder attempts to modify a critical system file, all modifications are saved in a second file and the original file remains unchanged. Additional modification attempts result in changes to the temporary shadow file and not the original file. Fisch proposed temporary shadow files as a

mechanism for protecting the integrity of a system while under active attack [10].

- Restrict User Activity: Suspicious users may be restricted to a special user shell that allows some functionality while limiting the ability of the user to execute certain commands. This will slow the user's ability to damage the system without terminating a user session, suspending user jobs, or requiring additional authentication.
- Logging to Unchangeable Media: One of the principal targets of any attacker is the system's log files. By logging to unchangeable media, the intruder cannot alter any evidence of the intrusion after it has been recorded. This is a viable tactic for preserving evidence.
- Process Accounting: Most systems do not routinely employ process accounting as it has high overhead and imposes a performance penalty on the host system. This is despite process accounting recording a plethora of useful information for diagnosing attacks. During an attack, however, the performance penalty is minimal compared to the utility of processing accounting and as such it is a viable tactic for collecting information on an ongoing attack.
- Employ a Honeypot: A honeypot attempts to attract the attention of the attacker so that the attack can be analyzed while protecting a critical system. High priority systems deploy honeypots all the time to divert attention away from their critical systems. Low priority systems can

deploy honeypots during an attack to redirect attention away from the protected system.

- Employ a Smokepot: A smokepot is a system on your network that any contact with is indicative of an attack. Smokepots are systems dedicated to detecting intrusions with no other function than to report contact. Like a honeypot, a smokepot helps detect attacks and can divert attacks away from production systems.
- Contact Servicing ISP: A tactic for responding to an attack is to contact the servicing internet service provider (ISP) and let the ISP respond to the attack.
- Turn off Modems: Turning off the modems will limit the ability of an attacker to reach and corrupt a system. While this might not stop the attack against an expert intruder, it will likely slow it again an expert and may stop a novice.
- Denial of Service (DOS) Attack: A tactics list would not be complete without the ability to attack back. DOS attacks are very expensive in terms of resources but can be effective in crippling the ability of an attacker to affect the protected system.
- System Compromise Attack: Similar to a DOS attack, a system compromise attack attempts to gain control of the attacker's system so that the attacker can no longer attack.

The relationship between plan steps and tactics are listed in Appendix B. The relationship between tactics and implementations are listed in Appendix C.

5. Response Taxonomy Component

The Response Taxonomy module receives input from the Analysis agent and determines an initial response weighting. It implements all of the dimensions of the Intrusion Response Taxonomy with the exception of the environmental constraints dimension which is implemented by the Policy Specification Component (see Section 6 below). In providing this classification, the Response Taxonomy component does not maintain state information - the Analysis component does. Everytime there is a new IDS report, the responsible Analysis component forwards all related state information for the Response Taxonomy module to reach a response goal classification. This state information consists of the previous classifications of the incident (history of type of attack, type of attacker, etc) as well as the current incident report.

6. Policy Specification Component

The Policy Specification module performs two functions: (1) it maintains any limitations on response goals and tactics; and, (2) it filters the plans and tactics generated by the Analysis and Tactics components. As discussed in Section B.6 of this chapter, not all responses are appropriate in all environments. The Policy Specification module provides a mechanism for restricting what responses are implemented in a given environment. These limitations include are legal, ethical, institutional, and resource constraints.

While environmental constraints are a critical component of any Response Taxonomy, the separation of policy specification module from the Response Taxonomy module in system design has two principal advantages. Policy specification is dependent on the environment in which the response system operates which can vary dramatically. For example, the response limitations of a small commercial, peacetime organization is significantly different than those of military organizations during a declared war. The other components of the taxonomy do not vary so widely. As such, the separation of the environmental constraints dimension of the response taxonomy from the other dimensions is preferred. Additionally, policy specification may change quickly while the other dimensions of the response taxonomy do not change over time. As new response resources are added to a system or new laws are approved, policy specification must change to reflect the operational environment of the response system.

The system administrator can use the System Administrator Interface to enter response limitations. The Policy Specification component returns a set of rules to the Analysis component that delimits appropriate response goals.

7. Response Toolkit Component

The Response Toolkit module is a collection of executables and system scripts that implement the intrusion response. These programs are system dependent and are invoked by the Tactics component. This separation of the Tactics and Response Toolkit component allows the proposed methodology to support multiple system architectures and provide a separation between the logic and implementation of the response plan. The

Response Toolkit also measures and provides feedback on the success or failure of any implementations.

8. System Administrator Interface

The System Administrator Interface module provides an interface for the system administrator to monitor and review incident and associated intrusion responses, suspend operation of the response system and assume an active role in the defense of the system, provide feedback to the system for adaptation, set system policy, and add new intrusion detection systems and associated interface components. The System Administrator interface receives reports from the Interface, Master Analysis, and Analysis components on incidents and associated responses. These events are correlated and displayed. After the security incident is resolved, the system administrator can indicate whether the intrusion was a real attack or a false positive report and whether the system response was successful. This allows the Interface component associated with a reporting IDS to update the confidence metric associated with the IDS and the Analysis and Tactics components to update their success metrics associated with various plans and techniques. The system administrator can also set system policy through the interface. These policy specifications are recorded in the Policy Specification component and are used to limit what responses the system implements.

D. Adaptation of Intrusion Response

The methodology provides response adaptation through two components: the Interface and Analysis components. The Interface component adapts by modifying the confidence metric associated with each IDS. After each incident, the system administrator can indicate whether an incident was an actual attack or a false positive report. This allows the system over time to adjust the response to an incident based on the system's confidence in the reporting IDS. Systems that habitually generate false alarms will result in less severe responses while systems that accurately detect intrusions will result in more robust responses.

The Analysis component also adapts to provide better intrusion response than non-adaptive systems. As the Analysis component receives additional incident reports, these reports may lead to reclassification of the attack. If significant changes are detected, replanning takes place to add additional PTI to the plan. By changing techniques with the same plan, the system adapts its approach in an attempt to stop the intruder. Finally, the Analysis component maintains success metrics on PTI. Those plans and actions that are successful in resolving intrusions are weighted so that they are used more frequently while those plans and actions that are not as successful are used less often.

E. Dealing with Uncertainty in Intrusion Response

The methodology addresses the inherent uncertainty of intrusion detection and response. IDSs are not perfect and will generate false reports. IRSs must adjust the response generated to the degree of certainty that the response system has that the intrusion detection report is valid. The confidence metrics, generated by the Interface components, provide a mechanism for mitigating the effect of uncertainty in IDSs.

The classification of incidents as either a new attack or an ongoing attack is likewise problematic. Attackers use a variety of techniques for masking their identity and these techniques are equally effective in foiling the classification of incidents as either new or an ongoing attack. An IRS must provide adequate protection to the system if this classification is incorrect by providing a gradual degradation in response effectiveness. The response to an incident must be sufficient to limit the effectiveness of the attack even if incorrectly characterized by the Master Analysis component. If incorrectly classified, the incident report is still forwarded to an Analysis module and the Analysis module still acts on the intrusive behavior. As such, while there is uncertainty in the Master Analysis component, this uncertainty does not invalidate the methodology.

The Response Taxonomy component also classifies incidents using several dimensions that involve uncertainty. Of the six dimensions of the response taxonomy, the system administrator specifies two dimensions (implications of the attack and environmental constraints) and these dimensions are assumed to be correct. A third dimension, timing, is easy to determine and does not involve any uncertainty. The fourth dimension, strength of suspicion, does involve uncertainty but this uncertainty is addressed by the confidence metric previously discussed. The number of attacks and the types of attacks is likewise used to develop the strength of suspicion. The final two dimensions, type of attacker and type of attack, involve significant uncertainty. The classification of type of attacker is dependent on and more difficult than the classification of a new or ongoing attack in the Master Analysis component. While the type of attack is founded on a sound theoretical model, there are no specific and universally accepted rules for attack classification. However, the longer the attacker is in the system, the more certain the IRS will become of the type of attacker and type of attack. Initial misclassification of the attacker and type of attack will not prevent the methodology from responding to the attack.

The response to an intrusion has one of three effects: stop the attacker; slow the attacker; or no effect. None of the responses in this research assist an attacker in corrupting a computer system. If the response using a misclassification of the attacker and/or type of attack stops the attacker, the misclassification had no effect and the issue of type of attacker and/or type of attack classification uncertainty is mute. If the response using a misclassification of the attacker/attack slows the attacker, the IRS will gain time to gather additional information which will lead to a better classification and effectively defend the computer system until the system administrator can take an active role in the defense of the system. In short, as long as the IRS stops or slows the attacker, the system is functioning properly.

If the response to the intrusion has no effect, then the proposed IRS has failed and uncertainty in the Response Taxonomy component may be factor. However, as soon as a new report is received, the response toolkit reevaluates the ongoing plan to determine its success. While there is uncertainty, this uncertainty is minimized as the system adapts it's plan of response.

The plan and tactics generated to respond to an attack will not be perfect and there will be some uncertainty as to which plan or tactic will be appropriate for a given response goal. The success metrics, generated by the Analysis component, and the reevaluation of the success of the plan after each report, provide mechanisms for limiting the effect of uncertainty in plan and tactics generation.

CHAPTER IV

IMPLEMENTATION

A. Introduction

In order to demonstrate the efficacy of this methodology it was necessary to implement a prototype based upon the design described in Chapter III. This chapter describes the implemented prototype. Each module of the methodology is described below. The Master Analysis, Analysis, and Taxonomy agent are the key components of this prototype and are discussed in great detail. The prototype components are depicted in Figure 17.

B. Implementation Overview

The system approach adopted in designing the agents was to opt for simplicity. The system has been designed to reduce the complexity of each agent and thus reduce the amount of information needed to perform a given task. Since each agent requires only a small amount of information, the communication between the agents is limited. Thus, the procedural-based approach using Java to perform all inter-agent communications is employed in this system.

This prototype is written in Java 1.2 and utilizes a Microsoft Access database for scenario and long-term system parameter storage. Special care was taken to isolate the scenario and IDS components of the system from the long-term system parameter storage so that future versions of AAIR could be linked to real instead of simulated IDSs and the system would continue to function correctly.

C. Intrusion Detection System Agent(s)

The intrusion detection systems in this prototype are simulated and are represented as Java threads. When the user loads a scenario from the AAIR Graphical

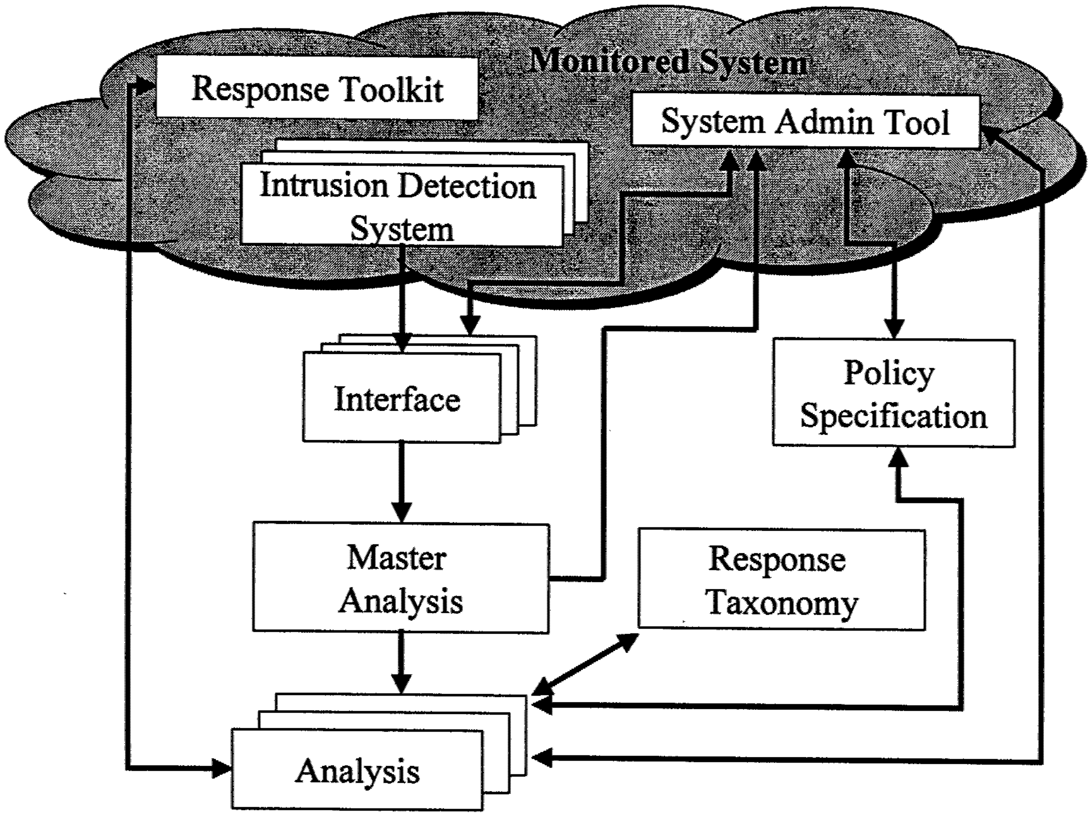


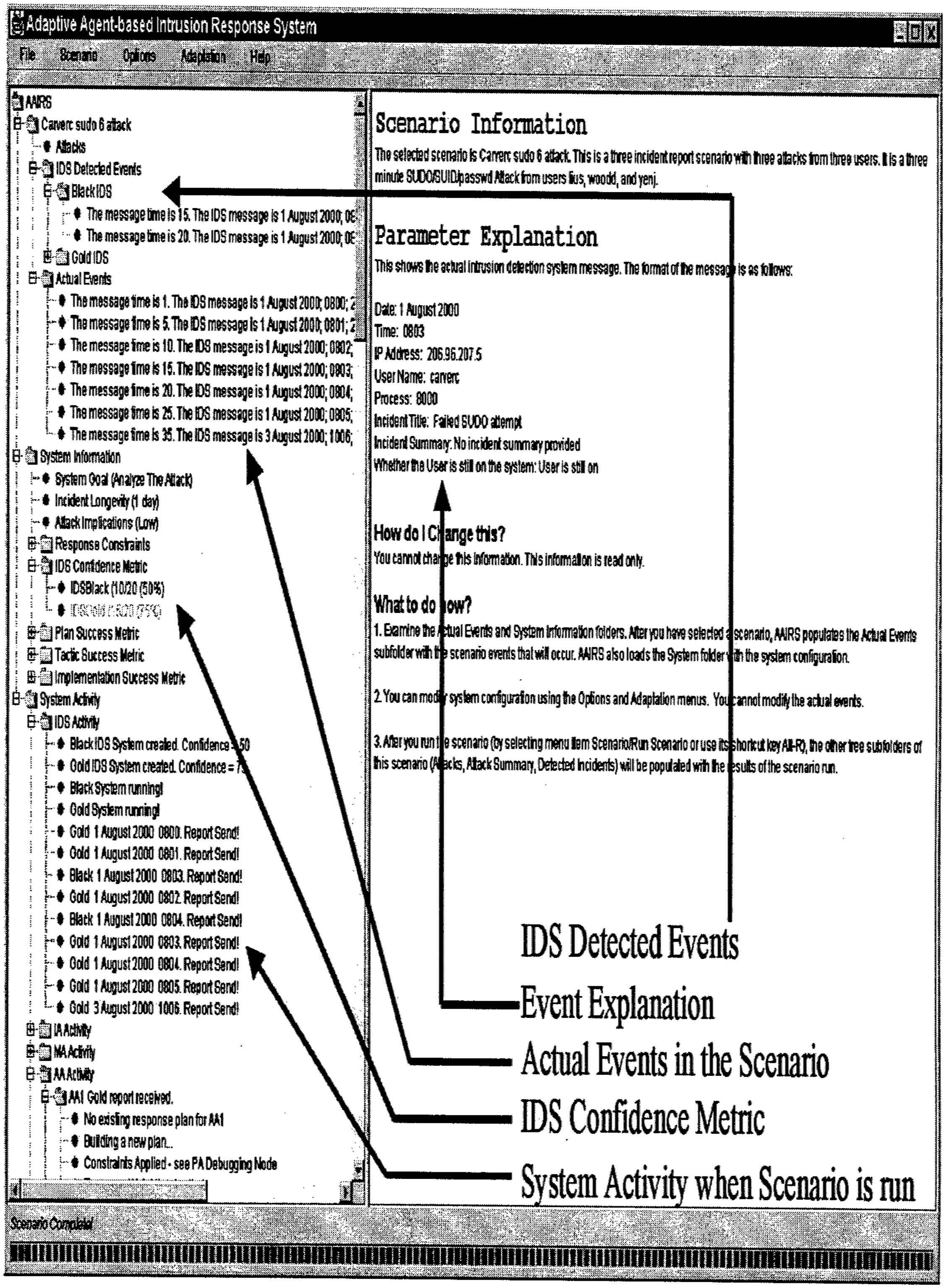
Figure 17: AAIR Prototype System

User Interface (GUI), all of the events in the simulation are loaded into an event list. A check is made for each event on the list to determine if the IDS detects that event. A second event list is built for each IDS containing only detected events using the IDS confidence. These detected events are the events that will drive the scenario. Additionally, as each intrusion detection system is created, it creates an associated interface agent to act as its buffer into the rest of the system. When the scenario is run, the IDS steps through its detected events list and forwards detected events to its interface agent.

Several IDS events are displayed in the user interface. Under the scenario information, the IDS detected events and the actual events in the scenario are displayed. As each event is clicked on, it is parsed into its component parts and displayed in the right text pane under parameter explanation. As the scenario runs, the IDS activity subtree is updated to depict events such as IDS agent creation and reports being sent to the appropriate Interface agents (Figure 18).

D. Interface Agents

The Interface agents (IA) handle communications with different intrusion detection systems (IDS) by parsing the incident report into component parts and maintain a model of each IDS based on the IDS confidence metric. The IDS confidence metric is the percentage of times the IDS has detected an intrusion previously to the total number of events. Each incident report has the following components:



- AARS
  - Carverc sudo 6 attack
    - Attacks
    - IDS Detected Events
      - Black IDS
        - The message time is 15. The IDS message is 1 August 2000; 08
        - The message time is 20. The IDS message is 1 August 2000; 08
      - Gold IDS
    - Actual Events
      - The message time is 1. The IDS message is 1 August 2000; 0800; 2
      - The message time is 5. The IDS message is 1 August 2000; 0801; 2
      - The message time is 10. The IDS message is 1 August 2000; 0802;
      - The message time is 15. The IDS message is 1 August 2000; 0803;
      - The message time is 20. The IDS message is 1 August 2000; 0804;
      - The message time is 25. The IDS message is 1 August 2000; 0805;
      - The message time is 35. The IDS message is 3 August 2000; 1006;
  - System Information
    - System Goal (Analyze The Attack)
    - Incident Longevity (1 day)
    - Attack Implications (Low)
    - Response Constraints
    - IDS Confidence Metric
      - IDSBlack (10/20 (50%)
      - IDSGold (15/20 (75%)
    - Plan Success Metric
    - Tactic Success Metric
    - Implementation Success Metric
  - System Activity
    - IDS Activity
      - Black IDS System created. Confidence = 50
      - Gold IDS System created. Confidence = 7
      - Black System running!
      - Gold System running!
      - Gold 1 August 2000 0800. Report Send!
      - Gold 1 August 2000 0801. Report Send!
      - Black 1 August 2000 0803. Report Send!
      - Gold 1 August 2000 0802. Report Send!
      - Black 1 August 2000 0804. Report Send!
      - Gold 1 August 2000 0803. Report Send!
      - Gold 1 August 2000 0804. Report Send!
      - Gold 1 August 2000 0805. Report Send!
      - Gold 3 August 2000 1006. Report Send!
    - IA Activity
    - MA Activity
    - AA Activity
      - AA1 Gold report received.
        - No existing response plan for AA1
        - Building a new plan...
        - Constraints Applied - see PA Debugging Node

Scenario Information

The selected scenario is Carverc sudo 6 attack. This is a three incident report scenario with three attacks from three users. It is a three minute SUDO/SUID/passwd Attack from users lius, woodd, and yenj.

Parameter Explanation

This shows the actual intrusion detection system message. The format of the message is as follows:

Date: 1 August 2000
Time: 0803
IP Address: 206.96.207.5
User Name: carverc
Process: 8000
Incident Title: Failed SUDO attempt
Incident Summary: No incident summary provided
Whether the User is still on the system: User is still on

How do I Change this?

You cannot change this information. This information is read only.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AARS populates the Actual Events subfolder with the scenario events that will occur. AARS also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. After you run the scenario (by selecting menu item Scenario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Detected Incidents) will be populated with the results of the scenario run.

Scenario Complete!

IDS Detected Events

Event Explanation

Actual Events in the Scenario

IDS Confidence Metric

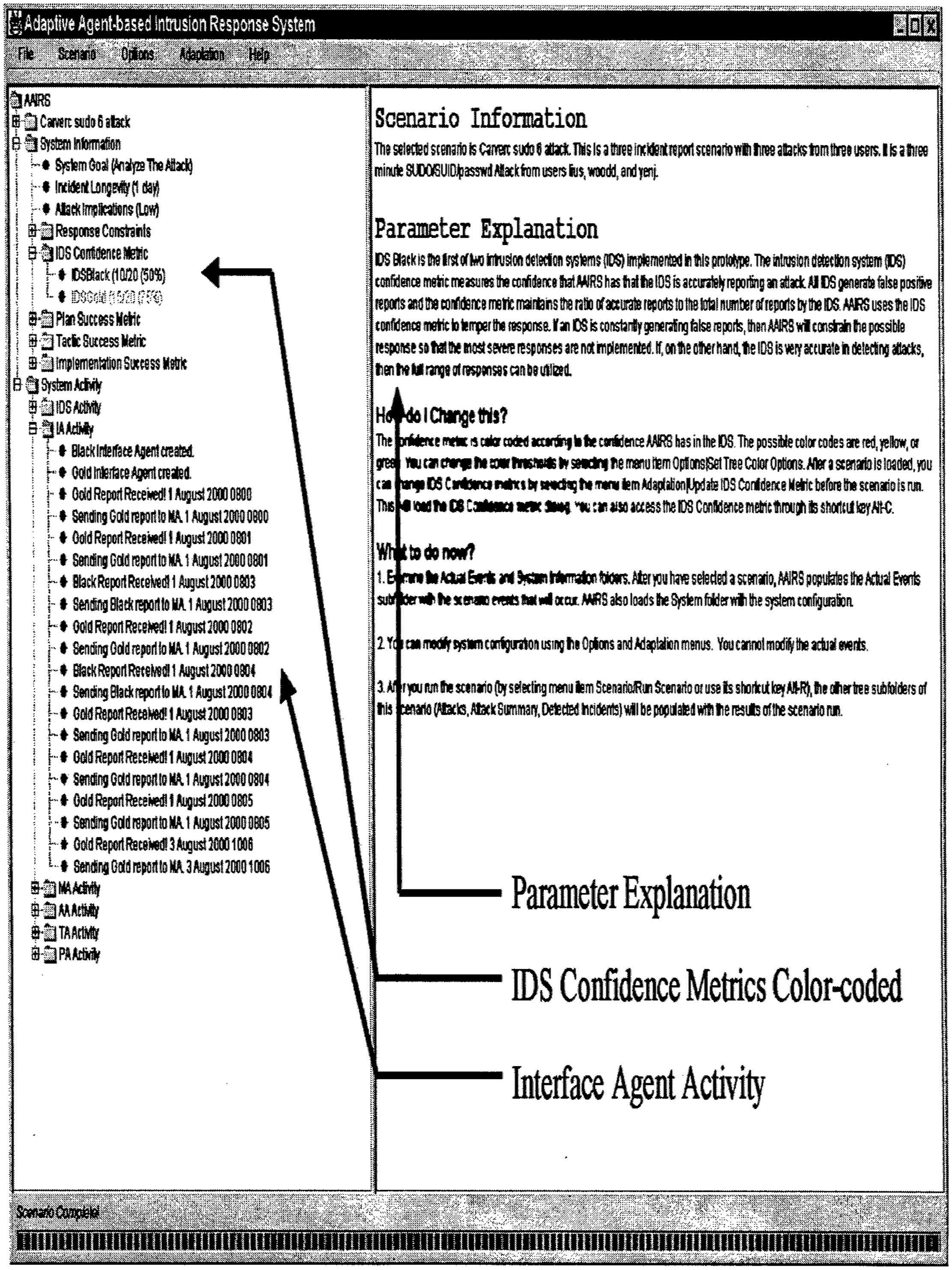
System Activity when Scenario is run

Figure 18: IDS Functionality

- Date/Time: The date and time of the incident report indicate when the intrusion was detected. This information is used by the Master Analysis agent to determine if the incident is part of an ongoing attack and to trim event histories that are passed to each Analysis agent.
- IP Address/Username/Process: The IP address, username, and process ID help AAIR identify the intruder. This information is used by the Master Analysis agent and Analysis agents for their internal metrics.
- Incident Title/Summary: The incident title and summary display the IDS classification of the intrusion and any supplemental information that the IDS has on the attack. This information is used by the Master Analysis and Response Taxonomy agents for their internal metrics.
- Whether the User is still on the System: Each incident report also has whether the user is still on the system. This information is used by the Response Taxonomy agent in weighting various PTI.

Each IA runs as a separate thread. The amount of delay between processing events for the entire system is controlled through the IA sleep function. The user can modify the simulation speed by setting the scenario delay under the menu item Scenario-Scenario Delay or through its shortcut key Alt-D. The default value is to process an event every 100 milliseconds although the user can select a value between 100 milliseconds and 5 seconds.

Several IA events are depicted in the AAIR GUI (Figure 19). Under system information, the IDS confidence metric is displayed and color coded according to the



- AAIRS
  - Carverc sudo 6 attack
  - System Information
    - System Goal (Analyze The Attack)
    - Incident Longevity (1 day)
    - Attack Implications (Low)
    - Response Constraints
    - IDS Confidence Metric
      - IDSBlack (10/20 (50%)
      - IDSGold (15/20 (75%)
    - Plan Success Metric
    - Tactic Success Metric
    - Implementation Success Metric
  - System Activity
    - IDS Activity
    - IA Activity
      - Black Interface Agent created.
      - Gold Interface Agent created.
      - Gold Report Received! 1 August 2000 0800
      - Sending Gold report to MA. 1 August 2000 0800
      - Gold Report Received! 1 August 2000 0801
      - Sending Gold report to MA. 1 August 2000 0801
      - Black Report Received! 1 August 2000 0803
      - Sending Black report to MA. 1 August 2000 0803
      - Gold Report Received! 1 August 2000 0802
      - Sending Gold report to MA. 1 August 2000 0802
      - Black Report Received! 1 August 2000 0804
      - Sending Black report to MA. 1 August 2000 0804
      - Gold Report Received! 1 August 2000 0803
      - Sending Gold report to MA. 1 August 2000 0803
      - Gold Report Received! 1 August 2000 0804
      - Sending Gold report to MA. 1 August 2000 0804
      - Gold Report Received! 1 August 2000 0805
      - Sending Gold report to MA. 1 August 2000 0805
      - Gold Report Received! 3 August 2000 1006
      - Sending Gold report to MA. 3 August 2000 1006
    - MA Activity
    - AA Activity
    - TA Activity
    - PA Activity

Scenario Information

The selected scenario is Carverc sudo 6 attack. This is a three incident report scenario with three attacks from three users. It is a three minute SUDO/SUID/passwd Attack from users lius, woodd, and yenj.

Parameter Explanation

IDS Black is the first of two intrusion detection systems (IDS) implemented in this prototype. The intrusion detection system (IDS) confidence metric measures the confidence that AAIRS has that the IDS is accurately reporting an attack. All IDS generate false positive reports and the confidence metric maintains the ratio of accurate reports to the total number of reports by the IDS. AAIRS uses the IDS confidence metric to temper the response. If an IDS is constantly generating false reports, then AAIRS will constrain the possible response so that the most severe responses are not implemented. If, on the other hand, the IDS is very accurate in detecting attacks, then the full range of responses can be utilized.

How do I Change this?

The confidence metric is color coded according to the confidence AAIRS has in the IDS. The possible color codes are red, yellow, or green. You can change the color thresholds by selecting the menu item Options|Set Tree Color Options. After a scenario is loaded, you can change IDS Confidence metrics by selecting the menu item Adaptation|Update IDS Confidence Metric before the scenario is run. This will load the IDS Confidence metric dialog. You can also access the IDS Confidence metric through its shortcut key Alt-C.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AAIRS populates the Actual Events subfolder with the scenario events that will occur. AAIRS also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. After you run the scenario (by selecting menu item Scenario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Detected Incidents) will be populated with the results of the scenario run.

Figure 19: Interface Agent Functionality

previous success of the IDS. The confidence metric can be modified by the user after the scenario loads through the menu item Adaptation-Update IDS Confidence Metric or its shortcut key Alt-C. The color-code thresholds can likewise be modified through the menu item Options-Set Tree Color Options. This allows users to customize the user interface. As each IDS is clicked on, it is parsed into its component parts and displayed in the right text pane under parameter explanation.

E. Master Analysis Agent

The Master Analysis (MA) agent classifies events as either part of an ongoing attack or as a new attack. To make this determination, the MA agent maintains an event list history for each Analysis agent and uses three internal metrics: time metric, session identifier metric, and attack type metric.

1. Event List History

The MA agent maintains an event history list for each Analysis agent (AA). While it was initially envisioned that this functionality would be provided by each AA, it became apparent that the MA agent had to have that functionality to complete its task. As such, the MA agent adds and deletes events from event lists. Events are added if the MA agent determines that the received report is a continuation of an ongoing attack. Events are removed when they are older than the incident longevity limit. The incident longevity is set by the system administrator and can be adjusted through the AAIR GUI menu Options-Set Incident Longevity.

2. Time Metric

The time metric evaluates the amount of time between the last received incident report for each Analysis agent and the current report. Time is classified as short, medium or long. If the difference between the two times is less than 10 minutes, then the time metric is set to short. If the difference is more than 10 minutes but less than 60 minutes, then the time metric is set to medium. If the time metric is longer than 60 minutes, then the time metric is set to long.

3. Session Identifier Metric

The session identifier metric looks at the IP address and user name to determine if the session information supports classifying the new report as either the continuation of an old attack or a new attack (See Table 4). The session identifier is classified as low, medium, or high and is a combination of the IP address and user name metrics. The IP address metric returns high if the IP address is the same. It returns medium if the IP addresses are different but part of the same subnet. It returns low if the IP addresses are from completely different networks. The user name metric returns high if the user name is the same on the two reports or low if the user name is different on the two reports. The

Table 4: Session Identifier Decision Table

Session Identifier	User	Result
High	High	High
High	Low	Medium
Medium	High	High
Medium	Low	Medium
Low	High	Medium
Low	Low	Low

Table 5: Master Analysis Agent Decision Table

Time	Session Identifier	Attack	Result
Short	High	High	Same
Short	High	Low	Same
Short	Medium	High	Same
Short	Medium	Low	Same
Short	Low	High	Same
Short	Low	Low	Different
Medium	High	High	Same
Medium	High	Low	Same
Medium	Medium	High	Same
Medium	Medium	Low	Same
Medium	Low	High	Same
Medium	Low	Low	Different
Long	High	High	Same
Long	High	Low	Different
Long	Medium	High	Same
Long	Medium	Low	Different
Long	Low	High	Different
Long	Low	Low	Different

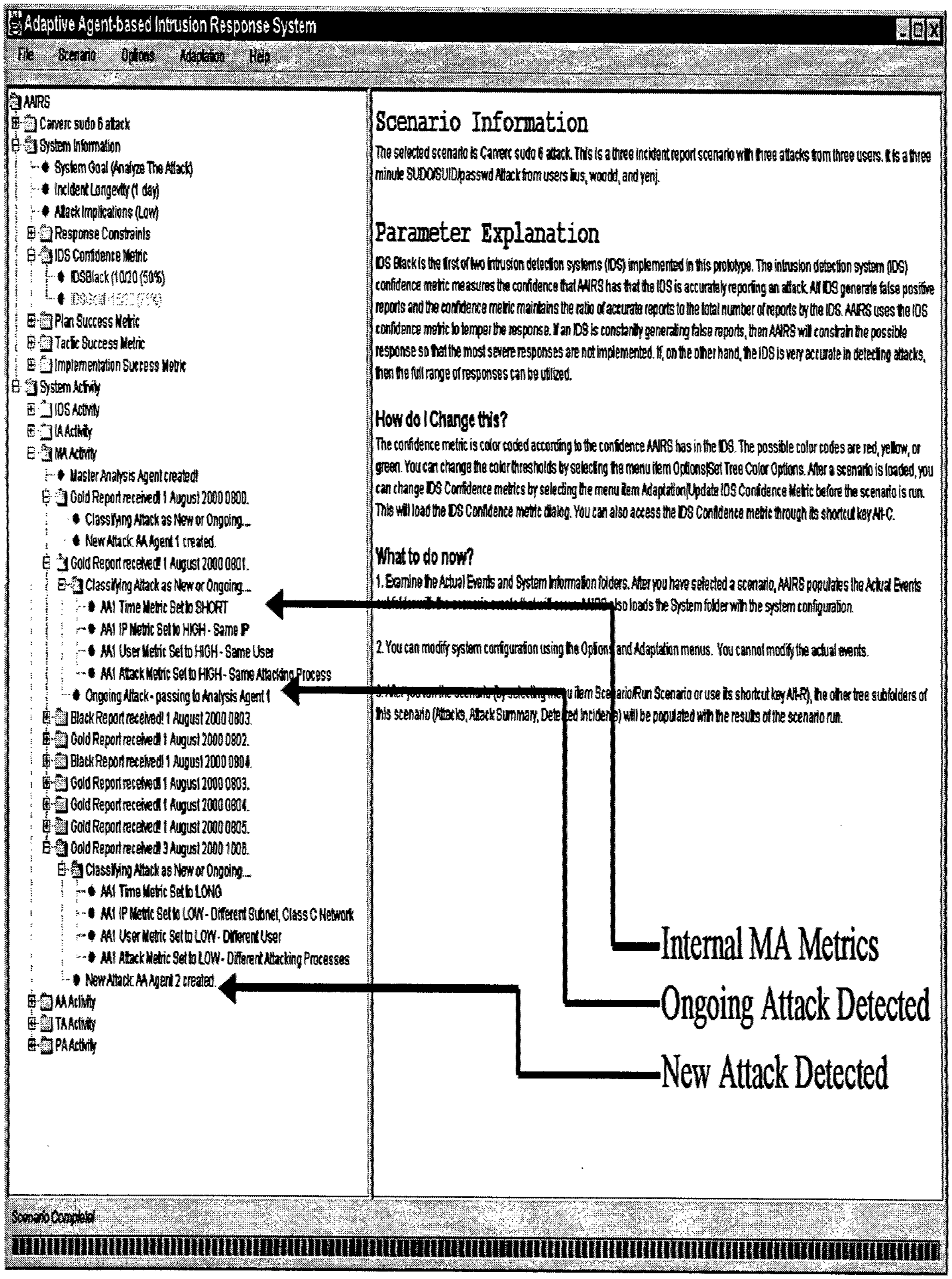
decision table for the session identifier metric is listed in Table 4 and the Master Analysis Decision Table is listed in Table 5.

4. Attack Type Metric

The attack type metric looks at the process initiating the attack and returns high if the attacking process is the same in the two incident reports or low if it is not.

5. Cumulative MA Decision-Making

The cumulative decision table is listed in Table 5. To determine the appropriate results for this classification, a survey of approximately ten security experts at Texas



- AAIRS
  - Carverc sudo 6 attack
  - System Information
    - System Goal (Analyze The Attack)
    - Incident Longevity (1 day)
    - Attack Implications (Low)
    - Response Constraints
    - IDS Confidence Metric
      - IDSBlack (10/20 (50%)
      - [illegible]
    - Plan Success Metric
    - Tactic Success Metric
    - Implementation Success Metric
  - System Activity
    - IDS Activity
    - IA Activity
    - MA Activity
      - Master Analysis Agent created!
      - Gold Report received! 1 August 2000 0800.
        - Classifying Attack as New or Ongoing...
        - New Attack: AA Agent 1 created.
      - Gold Report received! 1 August 2000 0801.
        - Classifying Attack as New or Ongoing...
          - AA1 Time Metric Set to SHORT
          - AA1 IP Metric Set to HIGH - Same IP
          - AA1 User Metric Set to HIGH - Same User
          - AA1 Attack Metric Set to HIGH - Same Attacking Process
        - Ongoing Attack - passing to Analysis Agent 1
      - Black Report received! 1 August 2000 0803.
      - Gold Report received! 1 August 2000 0802.
      - Black Report received! 1 August 2000 0804.
      - Gold Report received! 1 August 2000 0803.
      - Gold Report received! 1 August 2000 0804.
      - Gold Report received! 1 August 2000 0805.
      - Gold Report received! 3 August 2000 1006.
        - Classifying Attack as New or Ongoing...
          - AA1 Time Metric Set to LONG
          - AA1 IP Metric Set to LOW - Different Subnet, Class C Network
          - AA1 User Metric Set to LOW - Different User
          - AA1 Attack Metric Set to LOW - Different Attacking Processes
        - New Attack: AA Agent 2 created.
    - AA Activity
    - TA Activity
    - PA Activity

Scenario Information

The selected scenario is Carverc sudo 6 attack. This is a three incident report scenario with three attacks from three users. It is a three minute SUDO/SUID/passwd Attack from users lius, woodd, and yenj.

Parameter Explanation

IDS Black is the first of two intrusion detection systems (IDS) implemented in this prototype. The intrusion detection system (IDS) confidence metric measures the confidence that AAIRS has that the IDS is accurately reporting an attack. All IDS generate false positive reports and the confidence metric maintains the ratio of accurate reports to the total number of reports by the IDS. AAIRS uses the IDS confidence metric to temper the response. If an IDS is constantly generating false reports, then AAIRS will constrain the possible response so that the most severe responses are not implemented. If, on the other hand, the IDS is very accurate in detecting attacks, then the full range of responses can be utilized.

How do I Change this?

The confidence metric is color coded according to the confidence AAIRS has in the IDS. The possible color codes are red, yellow, or green. You can change the color thresholds by selecting the menu item Options|Set Tree Color Options. After a scenario is loaded, you can change IDS Confidence metrics by selecting the menu item Adaptation|Update IDS Confidence Metric before the scenario is run. This will load the IDS Confidence metric dialog. You can also access the IDS Confidence metric through its shortcut key Alt-C.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AAIRS populates the Actual Events [illegible] also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. [illegible] u item Sce[illegible]ario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Dete[illegible]ted Inciden[illegible]s) will be populated with the results of the scenario run.

Figure 20: Master Analysis Functionality

A&M and the United States Military Academy was conducted. The classifications were consistent so that a larger survey size was deemed unnecessary.

6. MA GUI

Several MA events are displayed in the AAIR GUI (Figure 20). As each report is received, it is classified as either an ongoing attack or new attack and the internal metrics (time, IP, user, and attack type) decisions are displayed. This allows the system administrator to easily trace decisions made by the MA agent.

F. Analysis Agent

The Analysis agent (AA) builds and implements a response plan. To build this plan, the AA takes the classification and incident report from the MA agent and invokes the Response Taxonomy and Policy Specification agents. It takes the classification and constraints from these agents and then uses its own internal logic to build the plan. This process is event-driven and the catalyst is the reception of an incident report. There are two possible scenarios: no plan exists and a new plan must be built; or, a plan exists and it must be checked for success and possible adaptation.

1. New Plan Generation

If the AA has not previously developed a response plan, it must devise one from scratch. It builds a new plan by initializing a plan array, applying policy constraints, setting response taxonomy weights, determining system response goal weights, building a tentative plan, and building a final plan. Each of these steps is discussed below.

a) Applying Policy Constraints: After initializing a plan array, a pointer to the array is passed to the Policy Agent. The Policy Array adds the policy constraints to the array and passes control back to the AA. At this point, the plan is constrained but no weightings have been applied.

b) Setting Response Taxonomy Weights: The Response Taxonomy agent takes incident report and classifies the attack. The details of how the report is classified are discussed in Section G below. The constrained plan is then weighted to reflect the response taxonomy classification.

c) Determining System Response Goal Weights: As discussed in Chapter III section C.4, the system response goal fundamentally affects which PTI are preferred. The response goal array is loaded with a weighting factor based on the response goal set by the system administrator. The weight matrix can be found in Appendix D.

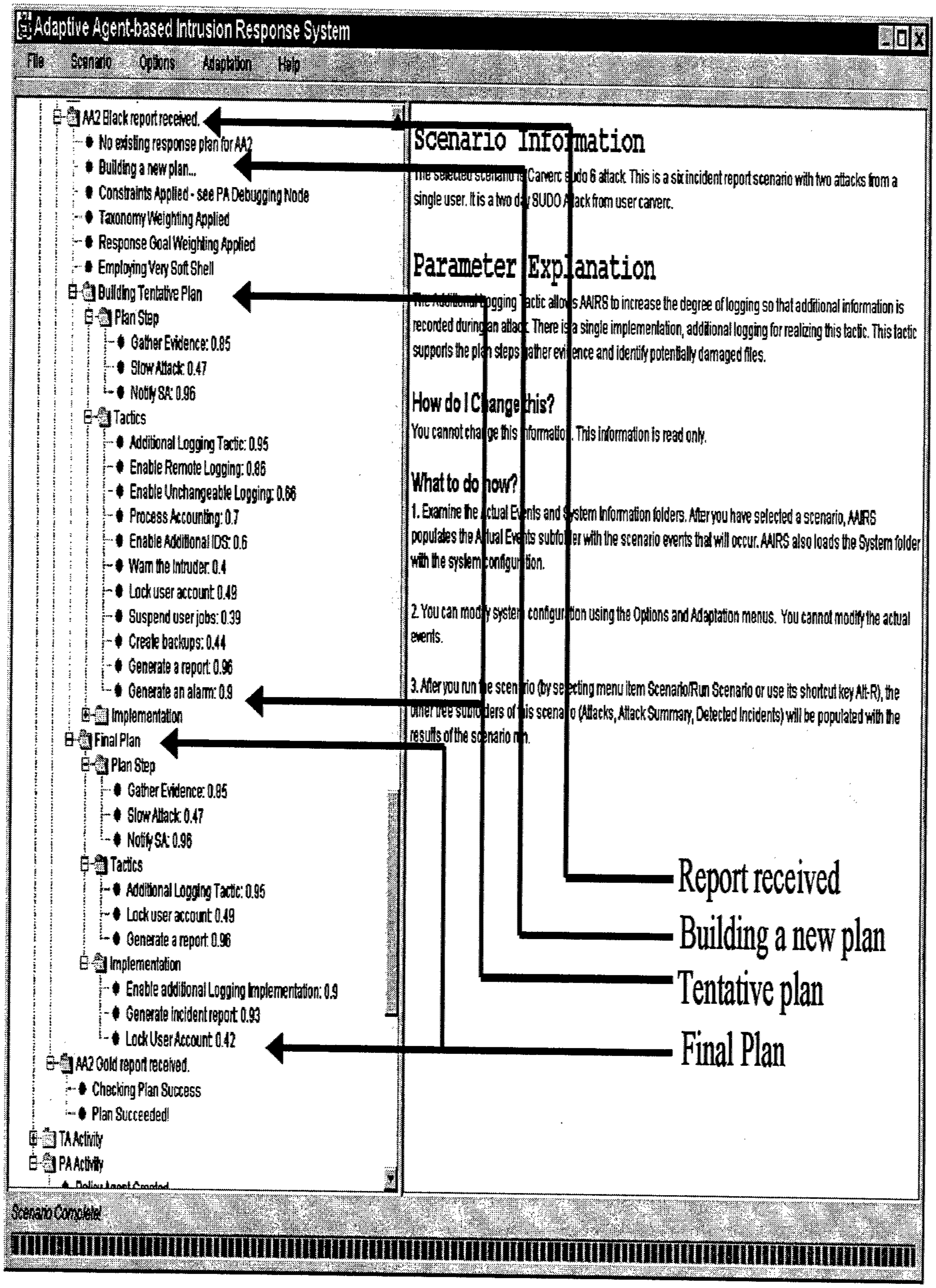
d) Building a Tentative Plan: The tentative plan lists all of the PTI that are viable for the plan. It makes this determination using the following formula:

$$TentPlan[i] = ((PTIArray[i] + ResponseWeights[i])/2)*success$$

where PTIArray[i] contains the policy constraints and response taxonomy weights, ResponseWeights[i] contains the response goal weights, and success is the previous success of that PTI. The result is an array, the tentative plan, with values between 0 and 1. The higher the value, the more appropriate the PTI is for the current situation. Each PTI is then checked for inclusion in the plan by rolling a random number. If the PTI value is higher then the random number, then it is viable for possible inclusion in the plan. This makes plan generation nondeterministic although those PTI that are more appropriate or more successful have a much higher probability of being in the final plan.

In building a response plan, it is important that the plan attempts to either slow or stop the intruder. Regardless of the attack, AAIR should notify the system administrator that the system is being attacked. Using the procedures listed above, it is possible that these plan steps are not viable due to the use of random numbers. As such, the tentative plan is checked to make sure that either the slow attack or stop attack plan step is viable and that the notify system administrator plan step is viable. The AA also checks to make sure that at least one supporting tactic and implementation for these plan steps is selected (See Appendix B and C for supporting tactics/implementations). The end result is a tentative plan that lists all of the PTI that are viable for the plan.

e) Building a Final Plan: The final plan takes the tentative plan and modifies it to reflect the relationships between PTI and reflect system criticality into the response (See Figure 21). While the tentative plan lists all viable PTI, it does not take into account the relationships between PTI. For example, the tentative plan may list the "gather evidence" plan step as being viable but there are no supporting tactics or implementations that support this plan step. The final plan reflects these relationships by recursively checking each viable plan step to make



```
AA2 Black report received.
    No existing response plan for AA2
    Building a new plan...
    Constraints Applied - see PA Debugging Node
    Taxonomy Weighting Applied
    Response Goal Weighting Applied
    Employing Very Soft Shell
    Building Tentative Plan
        Plan Step
            Gather Evidence: 0.85
            Slow Attack: 0.47
            Notify SA: 0.96
        Tactics
            Additional Logging Tactic: 0.95
            Enable Remote Logging: 0.86
            Enable Unchangeable Logging: 0.66
            Process Accounting: 0.7
            Enable Additional IDS: 0.6
            Warn the Intruder: 0.4
            Lock user account: 0.49
            Suspend user jobs: 0.39
            Create backups: 0.44
            Generate a report: 0.96
            Generate an alarm: 0.9
        Implementation
    Final Plan
        Plan Step
            Gather Evidence: 0.85
            Slow Attack: 0.47
            Notify SA: 0.96
        Tactics
            Additional Logging Tactic: 0.95
            Lock user account: 0.49
            Generate a report: 0.96
        Implementation
            Enable additional Logging Implementation: 0.9
            Generate incident report: 0.93
            Lock User Account: 0.42
AA2 Gold report received.
    Checking Plan Success
    Plan Succeeded!
TA Activity
PA Activity
```

Scenario Information

The selected scenario is Carverc sudo 6 attack. This is a six incident report scenario with two attacks from a single user. It is a two day SUDO Attack from user carverc.

Parameter Explanation

The Additional Logging Tactic allows AAIRS to increase the degree of logging so that additional information is recorded during an attack. There is a single implementation, additional logging for realizing this tactic. This tactic supports the plan steps gather evidence and identify potentially damaged files.

How do I Change this?

You cannot change this information. This information is read only.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AAIRS populates the Actual Events subfolder with the scenario events that will occur. AAIRS also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. After you run the scenario (by selecting menu item Scenario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Detected Incidents) will be populated with the results of the scenario run.

Figure 21: Building a New Plan

sure that it is supported by at least one underlying tactic and implementation. Those plan steps that are not supported are removed from the final plan. Additionally, the shut down host tactic and implementation are removed from the final plan if it had been selected as viable. Shutting down the host is a last resort tactic that should not be deployed if there are other viable tactics.

The final plan also incorporates system criticality through a "hard/soft" shell approach. Critical systems require a hard shell approach where a large number of PTI are deployed initially to protect the system. The system is critical and the response system is going to deploy most or all of the viable PTI to protect the system. For lower priority systems, a "softer" shell can be employed to protect the system. If these initial measures fail, the plan can be adapted to strengthen the defense shell. Table 6 lists the relationship between system criticality and the percentage of viable PTI deployed. The tentative and final plan are added to the plan history for future use in plan adaptation if adaptation is required. The tentative plan and final plan are displayed in the user interface so that the system administrator can review response plans.

Table 6: Relationship between System Criticality and PTI Deployment

System Criticality	Percent of PTI initially deployed
Low	0-25%
Medium	25-50%
High	50-75%
Critical	80-100%

2. Plan Adaptation

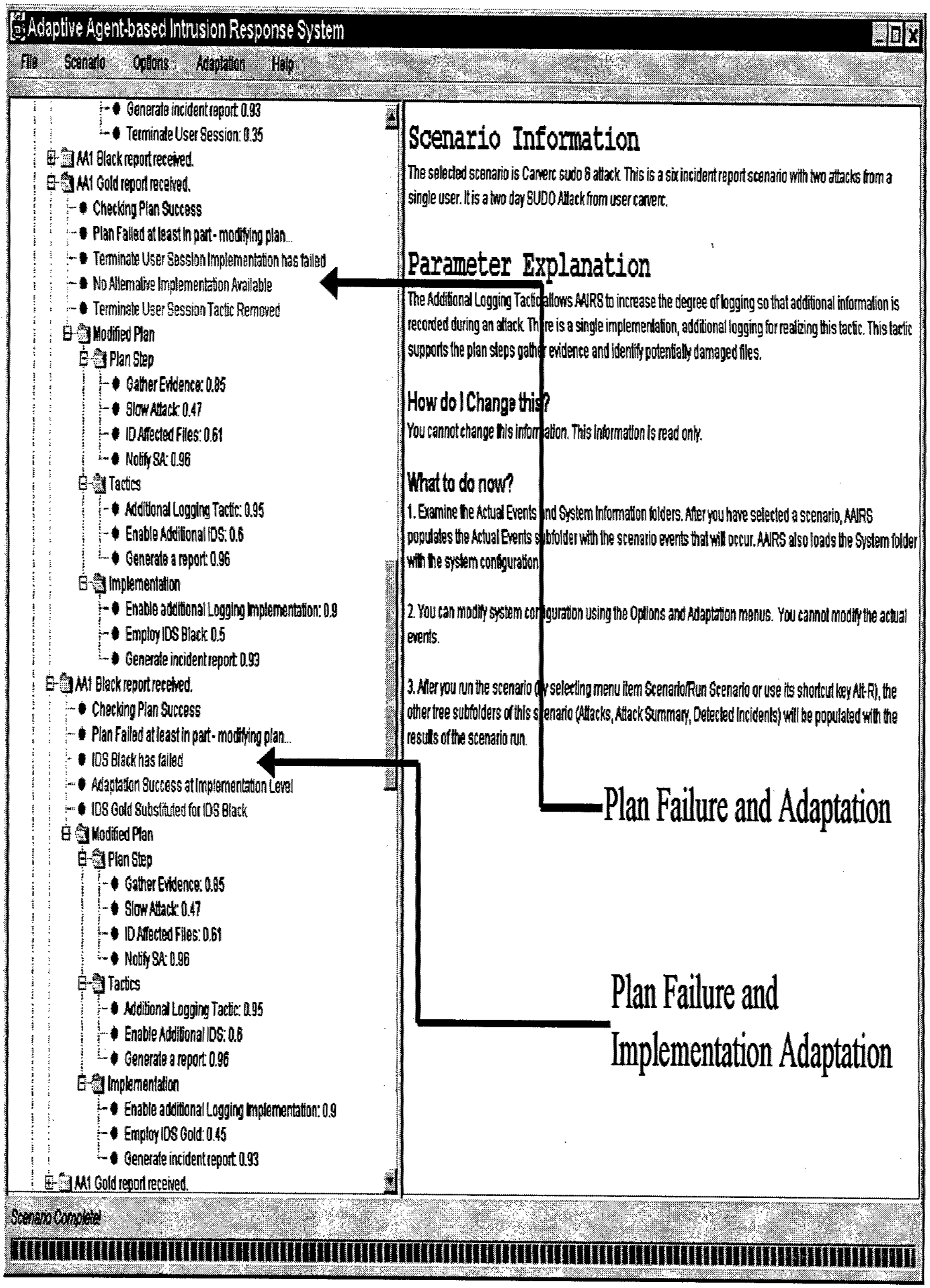
If there is an existing plan, the AA checks for plan success and significant changes in the environment. If there is a failure in the existing plan or significant changes in the environment, then the AA attempts to adapt the plan. Each of these situations is discussed below.

a) Plan Success: Each plan that is received is checked for plan success. The previous plan is loaded from the plan history and each implementation in the plan is checked for success by comparing its success to a random number. If the implementation's success is greater than the random number, then the implementation succeeded. If it is lower, then the implementation has failed and the plan has failed at least in part. Plan failure results in plan adaptation.

Plan adaptation starts at the implementation level and works up to the plan step level. Each failed implementation is checked to determine if there is an alternate implementation that has not previously failed and is not already in the plan. If there is an alternate implementation that is already in the plan, the failed implementation is simply removed from the plan. If there is a viable alternate implementation, it is added to the plan and the failed implementation is removed from the plan. If there is no viable alternative, the failed implementation and its tactic are removed from the plan.

If there is a failure at the tactics level, each of the plan steps are checked to ensure that they are still viable. If there has been a failure at the plan step level, a significant change has taken place and the AA develops a new plan, based on the old plan to remedy failures in the old plan. If all other tactics have failed, the AA instructs the response toolkit to shut down the host until the system administrator can take an active role in the defense of the system.

b) Significant Changes: If the previous plan succeeds, the AA checks to see if significant changes have occurred that may require adaptation. The AA builds a new plan based on the new incident report (See Section F.1 above) and then compares the new plan to the existing plan looking for significant changes in the tentative plans at the plan step or tactics level. A significant change is defined as a change of 0.3 or more in value. If a significant change is detected and the plan step or tactic is not in the existing plan already, the AA attempts to add the new plan step or tactic to the plan by checking for supporting tactics and implementations in the case of adding a plan step, or checking for supporting plan steps and implementations in the case of adding a tactic.



```
        • Generate incident report: 0.93
        • Terminate User Session: 0.35
AA1 Black report received.
AA1 Gold report received.
    • Checking Plan Success
    • Plan Failed at least in part - modifying plan...
    • Terminate User Session Implementation has failed
    • No Alternative Implementation Available
    • Terminate User Session Tactic Removed
    Modified Plan
        Plan Step
            • Gather Evidence: 0.85
            • Slow Attack: 0.47
            • ID Affected Files: 0.61
            • Notify SA: 0.96
        Tactics
            • Additional Logging Tactic: 0.95
            • Enable Additional IDS: 0.6
            • Generate a report: 0.96
        Implementation
            • Enable additional Logging Implementation: 0.9
            • Employ IDS Black: 0.5
            • Generate incident report: 0.93
AA1 Black report received.
    • Checking Plan Success
    • Plan Failed at least in part - modifying plan...
    • IDS Black has failed
    • Adaptation Success at Implementation Level
    • IDS Gold Substituted for IDS Black
    Modified Plan
        Plan Step
            • Gather Evidence: 0.85
            • Slow Attack: 0.47
            • ID Affected Files: 0.61
            • Notify SA: 0.96
        Tactics
            • Additional Logging Tactic: 0.95
            • Enable Additional IDS: 0.6
            • Generate a report: 0.96
        Implementation
            • Enable additional Logging Implementation: 0.9
            • Employ IDS Gold: 0.45
            • Generate incident report: 0.93
AA1 Gold report received.
```

Scenario Information

The selected scenario is Carverc sudo 6 attack. This is a six incident report scenario with two attacks from a single user. It is a two day SUDO Attack from user carverc.

Parameter Explanation

The Additional Logging Tactic allows AAIRS to increase the degree of logging so that additional information is recorded during an attack. There is a single implementation, additional logging for realizing this tactic. This tactic supports the plan steps gather evidence and identify potentially damaged files.

How do I Change this?

You cannot change this information. This Information is read only.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AAIRS populates the Actual Events subfolder with the scenario events that will occur. AAIRS also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. After you run the scenario (by selecting menu item Scenario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Detected Incidents) will be populated with the results of the scenario run.

Scenario Complete!

Plan Failure and Adaptation

Plan Failure and Implementation Adaptation

Figure 22: Plan Adaptation

In summary, the AA adapts the plan based on the current situation. If implementations in the plan fail, the AA attempts to replace the failed implementation and replans if there are no alternative implementations. If there is a significant change detected in the environment, the AA also attempts to replan and add new PTI to the plan to address the change. Figure 22 illustrates plan adaptation in the AA.

G. Response Taxonomy Agent

The Response Taxonomy (RT) agent classifies an incident to form the basis of a response plan. This classification is expressed as an array of PTI with corresponding weights. The weights are expressed as a number between 0.0 and 1.0. In performing this classification, the RT agent must determine the degree of suspicion, the time of attack, the type of attacker, the type of attack, and the attack implications. Time of attack and degree of suspicion are constraint criteria that remove PTI from possible inclusion in the plan. Type of attacker, type of attack, and attack implications are evaluative criteria that weight the PTI in terms of their suitability to the current situation. The array of PTI returned to the AA contains the average of the evaluative criteria with all constrained PTI removed from consideration. Each of these response taxonomy dimensions is discussed below in terms of how they are implemented in the prototype.

1. Degree of Suspicion

The degree of suspicion is determined with two metrics: incident count metric and incident type count metric. The incident count metric is a count of the incident reports that a particular AA has received. The more incident reports received, the greater the suspicion that the system is under an attack. The incident type count metric is a count of the different type of attacks on a system and like the incident count metric, the more types of attacks received, the greater the degree of suspicion. Both metrics return a number between 0.0 and 1.0. The formula for both metrics is:

$$incidentmetric = min(count/5,\ 1.0)$$

If either metric equals 1.0, then the suspicion is 1.0. Otherwise, the following formula is used:

$$Suspicion = confidence(incidentcount + typecount)/2$$

The suspicion metric is then used to constrain PTI. When the suspicion level is low, for example, twenty-four PTI are constrained so that they are not viable. When the suspicion level is high, no PTI are constrained. The complete suspicion metric table is listed in Appendix E.

2. Time of Attack

The time of attack metric is also used to constrain PTI. Certain PTI are not viable depending on when the attack occurred. For example, terminating a user session does not make sense after the user has left the system. The time of attack constraint table is listed in Appendix F.

3. Type of Attacker

The type of attacker metric classifies the attacker as either a novice or an expert and whether the attacker is launching a manual attack or an automated attack. To make these classifications, four metrics are employed. For human or automated, this method looks at the number of attacks as well as script attack patterns. If more than six incident reports are generated in less than a minute, then the attacker is classified as an automated attacker. Otherwise, the attacker is classified as a manual attacker. The script attack pattern method is not implemented in this prototype but should be a component in future prototypes.

For the classification of novice or expert, the system looks at the type of attack being used as well as the number of different attacks employed. Certain types of attacks are indicative of an expert attacker due to the complexity of the attack [36]. Additionally, experts can adapt their tactics to compromise a system and this will increase the number of different types of attacks.

Based on these four metrics, AAIR classifies the attacker as novice-automated, novice-human, expert-automated, or expert-manual. Based on that classification, different evaluative weights are assigned to each PTI. The complete type of attacker table is listed in Appendix G.

4. Type of Attack

The type of attack metric classifies the type of attack according to the Lindqvist intrusion result taxonomy [36]. There is a mapping between the incident title and the Lindqvist taxonomy classification. Different classifications result in different evaluative weights being applied to each PTI. The complete type of attack table is listed in Appendix H.

5. Attack Implications

The attack implications metric uses the criticality of the system to weight various responses. The system administrator sets the attack implications and then the TA does a table lookup. The complete attack implications table is listed in Appendix I.

H. Policy Specification

The Policy Agent (PA) records and applies policy and environmental constraints. The system administrator can set system constraints through the AAIR GUI using menu item Options-Response Constraints-Policy Constraints or its shortcut key Alt-Z (See Figure 23). Constraints can also be specified as a component of a scenario (discussed below). Constraints are set in the tentative plan array and remove the affected PTI for future consideration as a plan component.

I. Response Toolkit

The response toolkit executes implementations and monitors the success or failure of the implementations. This component is simulated. No implementations are actually executed. The success of the implementations is simulated through a random number generator that compares the random number with the success metric of a PTI. If the success metric is higher than the random number, then the PTI is deemed successful. In making this determination, only implementations are checked for success as plan steps and tactics are accomplished through implementations.

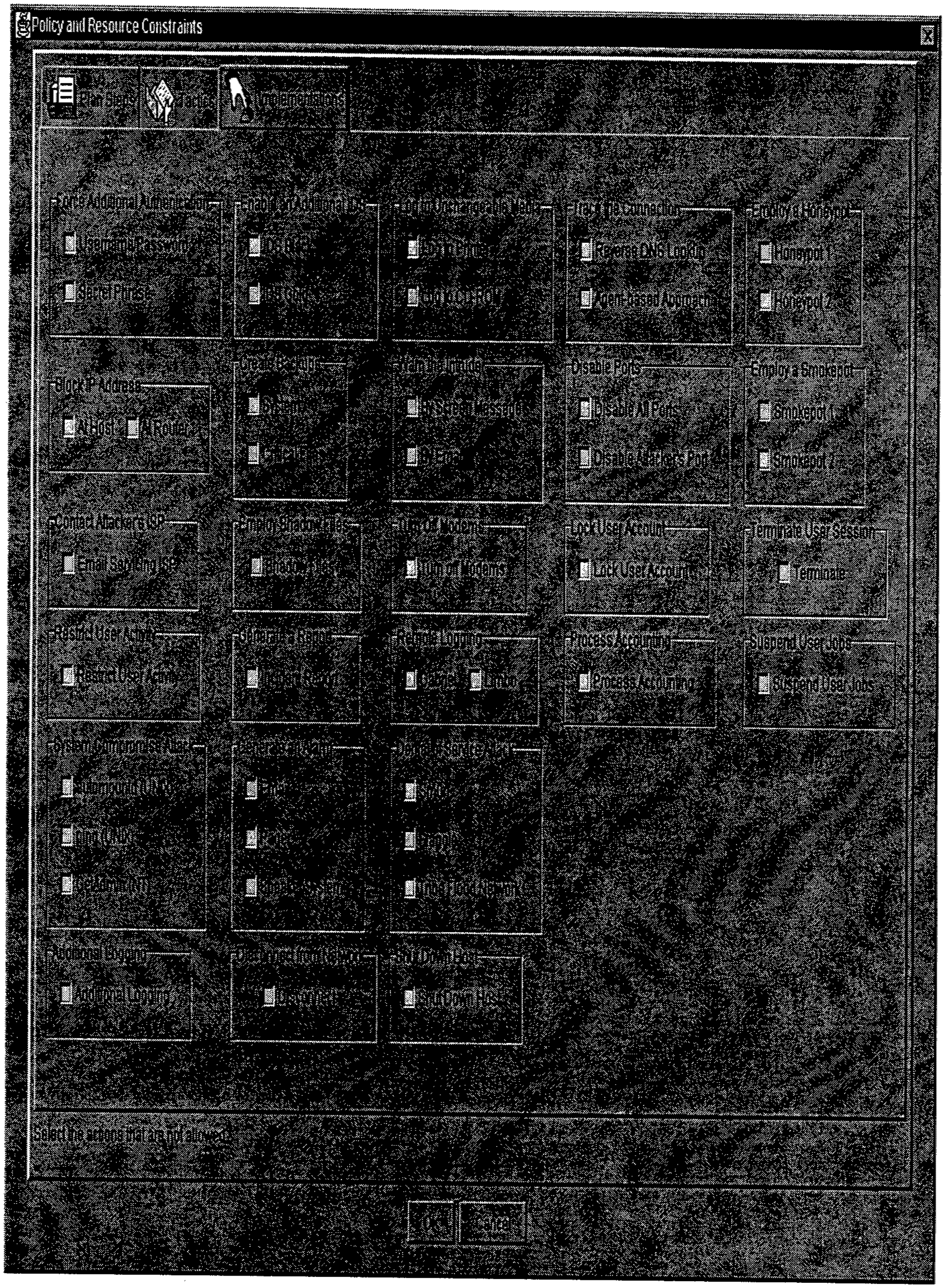


Figure 23: Policy Constraint GUI

J. System Administrator Interface

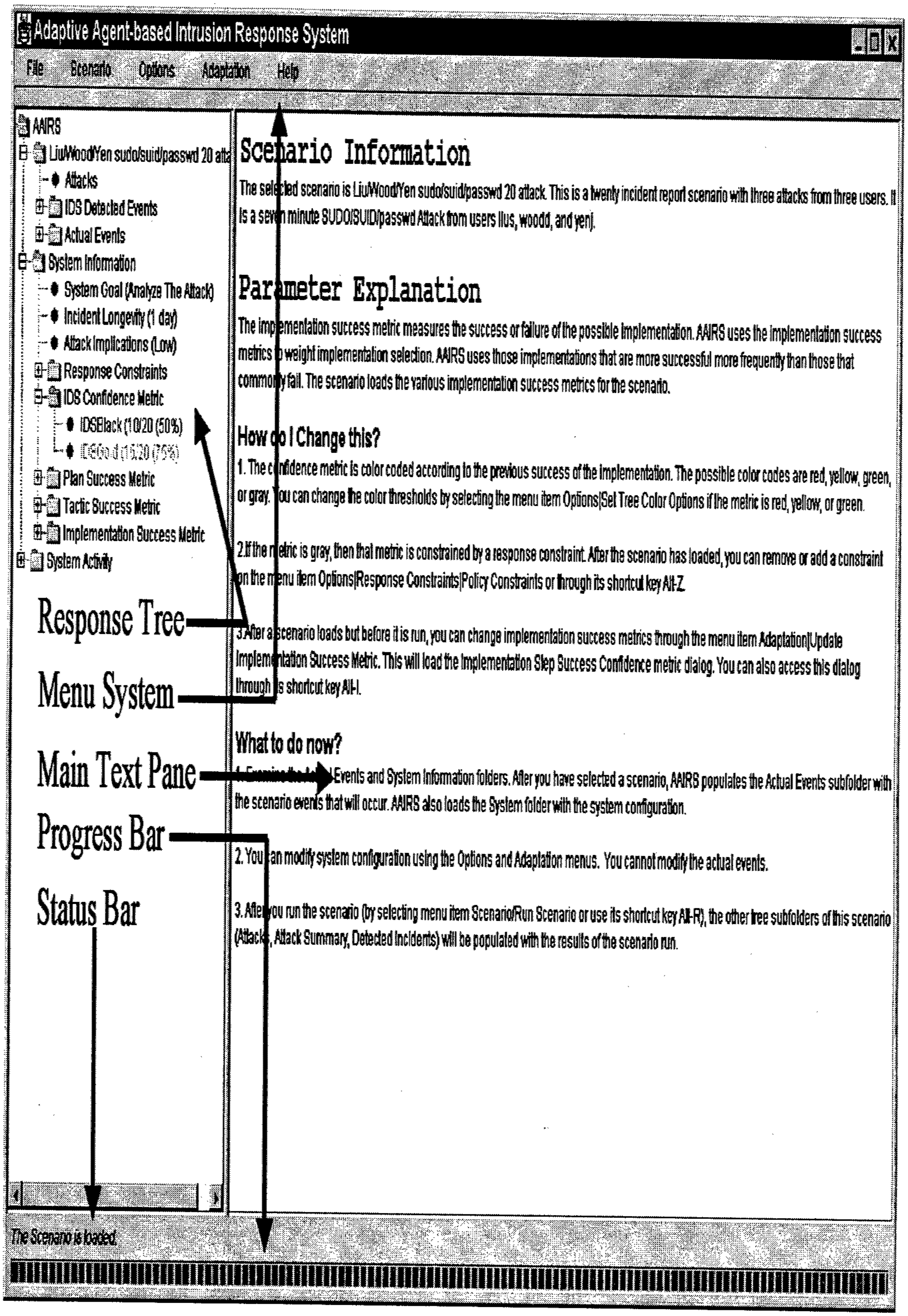
The system administrator interface allows the user to load and execute scenarios and monitor the AAIR response to scenario (See Figure 24). It consists of five principal components: a response tree, a menuing system, the main text pane, a status bar, and a progress bar. Each of these components are discussed below.

1. Response Tree

The response tree displays results of the response system. It is divided into three subtrees: scenario information, system information, and system activity.

a) Scenario Information: The scenario information subtree lists all scenario events as well as those that are detected by the IDSs in the scenario.

b) System Information: The system information subtree lists the system configuration for the scenario. This includes the system goal, incident longevity, attack implications, response constraints, IDS confidence metrics, and the PTI success metrics. The confidence and success metrics are color-coded red, yellow, or green according to the degree of previous success. The user can change the color code thresholds through the menu item Options-Set Tree Color Options and then clicking on a node in the tree. The user can also change any of the information in the system information subtree through the menu to tailor the scenario loaded.



Scenario Information

The selected scenario is LiuWoodYen sudo/suid/passwd 20 attack. This is a twenty incident report scenario with three attacks from three users. It is a seven minute SUDO/SUID/passwd Attack from users llus, woodd, and yenj.

Parameter Explanation

The implementation success metric measures the success or failure of the possible Implementation. AAIRS uses the implementation success metrics to weight implementation selection. AAIRS uses those implementations that are more successful more frequently than those that commonly fail. The scenario loads the various implementation success metrics for the scenario.

How do I Change this?

1. The confidence metric is color coded according to the previous success of the Implementation. The possible color codes are red, yellow, green, or gray. You can change the color thresholds by selecting the menu item Options|Set Tree Color Options if the metric is red, yellow, or green.

2.If the metric is gray, then that metric is constrained by a response constraint. After the scenario has loaded, you can remove or add a constraint on the menu item Options|Response Constraints|Policy Constraints or through its shortcut key Alt-Z.

3.After a scenario loads but before it is run, you can change implementation success metrics through the menu item Adaptation|Update Implementation Success Metric. This will load the Implementation Step Success Confidence metric dialog. You can also access this dialog through its shortcut key Alt-I.

What to do now?

1. Examine the Actual Events and System Information folders. After you have selected a scenario, AAIRS populates the Actual Events subfolder with the scenario events that will occur. AAIRS also loads the System folder with the system configuration.

2. You can modify system configuration using the Options and Adaptation menus. You cannot modify the actual events.

3. After you run the scenario (by selecting menu item Scenario/Run Scenario or use its shortcut key Alt-R), the other tree subfolders of this scenario (Attacks, Attack Summary, Detected Incidents) will be populated with the results of the scenario run.

The Scenario is loaded.

Figure 24: AAIR GUI Components

c) System Activity: The system activity subtree depicts the actual running of the system (See Figure 24). The IDS, interface agents, master analysis agent, analysis agents, taxonomy agent, and policy agent all have folders that display the internal operations of the system. The results of all of the metrics previously mentioned are displayed so that the system administrator can trace the actions of the response system.

2. Menuing System

The AAIR menuing system provides access to all of the response system's functions. Users can open, edit and run scenarios and watch AAIR response to incident reports. Menu items have popup help associated with them so that if the user hovers his mouse over the menu item, a short description of menu item's functionality is displayed in a popup window. The most commonly used menu items have shortcut or accelerator keys assigned to them to facilitate rapid manipulation of the system.

3. Main Text Pane

The main text pane provides detailed information on the tree node selected. It consists of four components: scenario information; parameter explanation; how do I change this; and, what to do now.

a) Scenario Information: This section of the main text pane displays a short description of the loaded scenario. If no scenario is loaded, the scenario information indicates no scenario has been loaded.

b) Parameter Explanation: Each node in the response tree has an associated explanation so that when the user clicks on a node, an explanation of the node contents is displayed under parameter explanation.

c) How Do I Change This: If the selected node's information can be changed, this portion of the main text pane displays the procedures for changing the node's information.

d) What to Do Now: This portion of the main text pane displays what the user should do next. If a scenario is not loaded, the system displays how to load a scenario. If a scenario is loaded but has not been run, the system displays how to change the scenario parameters and how to run a scenario. The advice provided is node sensitive so that as the user changes nodes in the response tree, the advice provided changes.

4. Status Bar

The status bar provides immediate feedback to the user as the system executes and key events occur.

5. Progress Bar

The progress bar provides a graphical representation of the system's progress in completing a task. This is primarily used to indicate progress in running a scenario.

K. Scenario Management

Scenarios are a collection of incident reports and a system configuration that allow the user to see AAIR in action. Scenarios are stored in a series of Microsoft Access tables. The incident reports are loaded into internal AAIR tables and drive the IDSs. The system configuration consists of IDS confidence metrics, PTI success metrics, and constraints as well as system parameters such as the response goal and incident longevity. Approximately four hundred variables must be loaded for each scenario run.

L. Summary

The prototype system includes a response taxonomy and implements the proposed adaptive intrusion response methodology. The Interface Agents receive incident reports and forward those reports along with a confidence metric to the Master Analysis Agent. The Master Analysis Agent classifies the incident as a new attack or as part of an ongoing attack and forwards the incident report to the appropriate Analysis Agent. The Analysis Agent builds a response plan to handle new attacks or adapts existing plans if the incident is part of an ongoing attack. In devising plans, the Analysis Agent invokes the Response Taxonomy Agent and Policy Agent. The prototype agents use a combination of fuzzy rule-bases, crisp rule-bases, and utility functions to make decisions and respond effectively to intrusions. The attached CD-ROM contains the code for the prototype system as well is a user's manual.

CHAPTER V

RESULTS

A. Introduction

This chapter describes the results of this research. Five domain experts were shown the prototype and their feedback was used to evaluate the methodology and taxonomy. The process used is described below as well as the results.

B. Comparison to Other Systems

There are no existing systems with the capabilities implemented in AAIR. As discussed in Chapter II, there are no systems that provide adaptive response or that implement a response taxonomy. Current response systems provide a static defense with all response adaptation provided by the system administrator. Since there are no existing systems with which to compare the methodology in this research, an alternate method of evaluation was used.

C. Experiments

As discussed in Chapter IV, a prototype system was built to demonstrate the feasibility of the methodology. Once the prototype was completed, several experiments were conducted with the purpose of verification and validation. These experiments were implemented as scenarios. In all these experiments, 1-3 attackers were simulated attacking a single system that employed two IDSs to protect the system.

The first set of experiments involved testing the classification of attacks as an ongoing or new attack. Twelve scenarios representing attacks by one, two and three different attackers were conducted. The length of each attack was also varied from three incident reports to twenty incident reports with the system performing as expected and classifying the attacks correctly.

The second set of experiments tested the plan generation portion of the system. A series of scenarios were developed in which the system developed a response plan. These scenarios varied from simple repeated SUDO attempts to sophisticated attacks that consisted of three different ongoing attacks each employing multiple attack techniques. The security experts examined the situation and the response plan and universally confirmed that the response plan was viable and appropriate.

The third set of experiments involved testing the adaptive nature of the system. A series of scenarios were developed in which the system had to adapt due to plan failures. These plan failures required adaptation at the tactics and implementation level as well as adaptation based on significant changes in the environment. The security experts examined the situation and the resultant system adaptation and again concluded that the adaptation was viable and appropriate.

D. Verification

Verification is the process of ensuring that the system performs as designed ("Is the code correct?") based on software engineering specifications. This is essentially an issue of software writing and testing. Does the code do what the designer wanted it to do? Techniques for verifying a system include [47]:

- Write and debug the program in subprograms or modules.
- Have multiple external people review the programming.
- Run the program under a variety of inputs and see if the output is reasonable.
- Use an interactive debugger or print out program traces to ensure each component of the system performs correctly.
- Run the system under simplifying assumptions.
- Examine animations of the program output.

Banks, et al., provide a similar list [48]. However, Banks recommends a graphical interface for accomplishing verification and validation due to its usefulness as a form of self-documentation.

Four of the previously mentioned techniques were employed in the verification of AAIR. Modular programming using an object-oriented approach facilitated the development of small, easily verifiable system components. Objects could be isolated and tested so that the functionality of each object could be verified independently of the other objects in the system. Verification was also accomplished by having two external programmers and security experts examine the program code. This periodic review of code by subject area experts who were fluent in Java, in many cases, led to significant enhancements to the program and verification that the program was working as designed and was correct.

The two remaining verification techniques employed running the program under a variety of input parameters to check if the output was reasonable and by animated traces of program execution. By running the program using a variety of input parameters, boundary conditions could be checked and verified. By having security experts check system output under a variety of situations and conclude the system output is reasonable, the overall correctness of the system can be verified. Finally, the contents of the system activity subtree in the AAIR GUI represents a very detailed and extensive trace of the system. All of the internal metrics are visible and the user can easily confirm that the system is working as designed and specified.

Given the reliance of AAIR on crisp rule bases, testing these rule bases was critical to the verification of the entire system. Fortunately, the crisp rule bases were small. Several test cases were built. The facts that corresponded to these test cases were asserted into the rule base. The output of the crisp rule base was compared to the known correct answer for the test case. Testing was conducted of the rule bases in the MA agent, AA, and TA. The various agents and their associated rule bases performed as specified.

E. Validation

Validation is the process of ensuring that the program solves the desired problem. Unlike verification, validation is used to determine whether the designed system is useful to the target consumer. Validity is also used to decide whether the system provides answers that "make sense" or are useful to human experts. Validation was done via extensive testing and experimentation, based on input from security experts. Law and Kelton provide a three-step approach for providing validation [47]:

- Face validity
- Test the assumptions of the system empirically
- Determine how representative the output data is

Due to lack of systems with which to compare the proposed methodology, the principal method of system validation was face validity. Face validity was obtained by placing security experts in front of the system and collecting their assessment of how well the system performed.

1. Validation of the Master Analysis Agent

The validity of the MA agent was demonstrated by prototype system AAIR and the first experiment on the prototype system. Given an incident report, the MA agent correctly classified the incident either as a new attack or as part of an ongoing attack. The sub-metrics internal to the MA agent performed as expected and were validated by the security experts. These experts also validated that the classification was useful in forming a response plan.

2. Validation of the Analysis Agent

The validity of the AA was demonstrated by the prototype system and the second and third experiment. Given a set of incident reports, the AA develops a response plan and then intelligently adapts that plan as the situation changes. When a plan component failed, the AA removed the failed PTI and either substituted an alternate implementation or replanned at the tactics level. If there is a significant change in the environment, the AA adapts the plan by adding the appropriate PTI. The AA solves the real-world problem of adaptively responding to an intrusion.

3. Validation of the Response Taxonomy

The validity of the TA was demonstrated by the prototype system and the second and third experiment. The success of the AA is dependent on the success of the TA in weighting the various PTI given a situation. The AA cannot succeed if the TA fails. Given that the AA is valid, it follows that the TA must likewise be valid.

CHAPTER VI

SUMMARY AND CONCLUSIONS

A. Summary

The purpose of this research was to develop a taxonomy and methodology for intrusion response. The taxonomy provides a theoretical foundation for the consistent classification of intrusion incidents and the selection of intrusion response goals. The methodology provides an adaptive approach for providing intrusion response that significantly extends the state of the art in intrusion response. The methodology replaces the limited decision tables found in most IRSs with a robust and explicit reasoning mechanism that adapts over time to provide better intrusion response.

The implemented prototype demonstrates that the overall methodology is sound. During an attack, the AAIR system effectively responds to an incident by developing a response goal, developing a plan to obtain that response goal, and implementing that plan. AAIR adapted the plan over the course of the incident based on the success or failure of the initial plan. The system also effectively adapted responses over the long terms based on the confidence and success metrics maintained by the system. The testing of this system indicates that AAIR is working as designed and implemented.

B. Significance of Research

The principle contribution of this research is to provide a taxonomy and methodology for intrusion response. This research addressed a number of important issues:

- The development of an intrusion response taxonomy. This research significantly extends previous security response taxonomies to include a number of new taxonomy dimensions such as type of attacker, type of attack, and environmental constraints. The taxonomy provides the theoretical infrastructure necessary for any intrusion response system.
- The development of an intrusion response methodology with explicit reasoning mechanisms. This research significantly extends previous intrusion response systems by providing a sound methodology for reasoning and responding to intrusions. Instead of using limited decision tables, the AAIR methodology provides a cognitive framework for responding coherently and adaptively to intrusions.
- Response adaptation to intrusive behavior. The system adapts over time to provide better responses by modeling associated intrusion detection systems and the success and failures of plans and tactics. As attackers change their exploitation techniques, the response systems adapts so that it can better thwart their attacks.

The overall benefit of this research is to provide a foundation on which other intrusion response systems can be built. This methodology will support automatic, adaptive intrusion response that will limit the effectiveness of computer attackers. The implemented prototype demonstrates the feasibility of this approach. Given a production system with the capabilities of the prototype system, a system administrator will be more effective in defending a computer system from attack.

C. Future Work

The prototype AAIR was never intended to be a commercial system. It was intended to demonstrate the feasibility of the overall methodology. The area of automatic intrusion response is an emerging research area that will require extensive research. As such, there are areas of future work to be explored in the AAIR prototype.

1. Development of the Response Toolkit

The prototype did not implement the response toolkit as the actual responses were not critical to an evaluation of the methodology. However, the implementation of a modular response toolkit would provide a suite of response tools that could be used by system administrators, IDSs, and IRSs. It would facilitate the inclusion of both manual and automatic intrusion response capabilities in future security tools and provide a standard implementation for responding to attacks. This would be an outstanding Master's degree project or thesis.

2. Interface Agents

Maintaining multiple confidence metrics on the same IDS instead of a single metric should enhance the accuracy of the IDS models within the interface agents (See

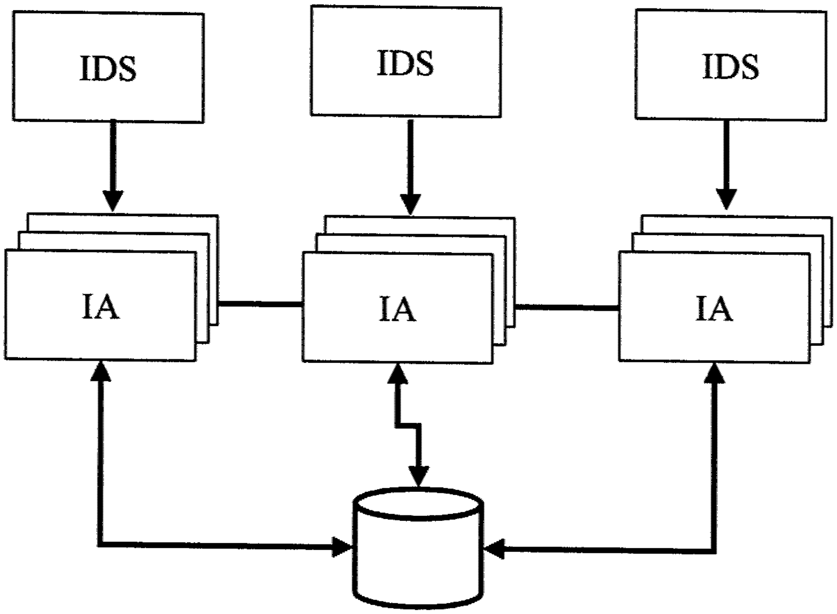


Figure 25: Enhanced Interface Agent Architecture

Figure 25). For example, an IDS may provide excellent detection of buffer-overflow attacks and poor detection of race condition attacks. This increase in model granularity may be worth the performance and complexity cost of implementing and maintaining multiple confidence metrics. Additional research is necessary.

The interface agent is an obvious, high priority attack candidate. If the interface agents stop functioning, the system cannot respond to computer attacks. Adding redundant interface agents to the same IDS such as indicated in Figure 25 would enhance the survivability of the response system.

3. Network Support

The methodology provides an intrusion response system for a single computer system. A distributed IRS system would offer significant advantages. Agent components of the system could be mobile and move from system to system thus enhancing their survivability. Intrusion responses could be coordinated across multiple systems so that the attacked system was not necessarily the system to enact the response. Dedicated and hardened "bastion systems" could respond to attacks thus drawing future attacks to themselves instead of computing systems. Network support is the next logical step in the research of IRSs.

The methodology could provide network support with the following modifications (See Figure 26). Each computing system would require a coordination

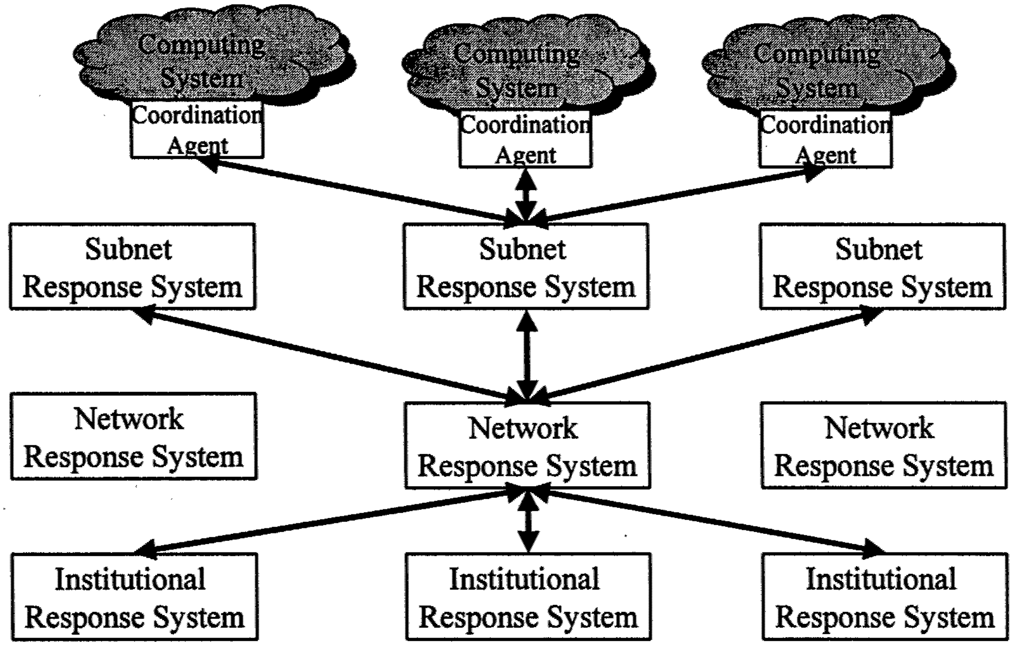


Figure 26: Network Architecture Extension

agent to handle communications with the response component of other systems. Response systems would be organized hierarchically in a three-tier architecture system with subnet, network and institutional response systems. The response to an intrusion is performed at the lowest applicable level while information sharing occurs at all levels to ensure that all components of the response system are aware of ongoing incidents. The functionality internal to each response system (subnet, network, and institutional) would remain as in this dissertation with the exception that a subordinate system may be directed to implement a response from a response system higher in the response hierarchy.

4. Agent Protection

Any IRS is a primary attack candidate. If an attacker can corrupt the response system, he or she has a perfect vehicle for attacking other systems. This prototype did not address this issue. Future response systems should incorporate internal protection mechanisms so that the response system cannot be compromised. An authorization and authentication infrastructure based on public key technology such as the one currently being researched by Humphries [49] appears to be promising for addressing the issue of agent protection in intrusion response systems.

5. Better User Interface

While the user interface is adequate for a prototype, user interface design for an IRS is an area of future research. The interface of a real-time system is a fundamental determinant of the system's effectiveness. The interface must be intuitive, easy to use, and powerful. It must present the right information at the right time for the system administrator to effectively monitor and affect an ongoing attack. The system administrator must be able to rapidly assess what actions the response system has taken, what responses are currently being implemented, the effectiveness of these actions. The system administrator should be able to quickly choose a mix of manual and automatic responses to a system with the system adapting based on the user decisions. After an attack, the system administrator must be able to assess the damage to the attacked system from the same interface. The system administrator must be able to rapidly select the degree of detail necessary to make decisions. This information presentation ranges from the abstract to the specific. Reporting and long-term analysis of intrusion responses is similarly a necessary component. While the current interface is not ineffective, further work is necessary to fine-tune the efficacy of this component.

6. Long-term Adaptation to Known Attackers

The methodology provides adaptive intrusion response throughout a session. It does not provide, however, for the maintenance of state information and attack signatures of known attackers. By maintaining this information, the analysis agent could tailor its response plan to the attacker and provide an implementation mechanism for a long-term plan to detect and defeat known attackers.

REFERENCES

[1] CERT Coordination Center, "CERT/CC Statistics for 1988 through 1998," Available at http://www.cert.org/stats/cert_stats.html, January, 2000.

[2] CERT Coordination Center, "CERT Coordination Center 1998," Available at http://www.cert.org/annual_rpts/cert_rpt_98.html, January, 2000.

[3] CERT Coordination Center, "Results of the Distributed-Systems Intruder Tools Workshop," Available at http://www.cert.org/reports/dsit_workshop.pdf, March 2, 2001.

[4] F. B. Cohen, "Simulating Cyber Attacks, Defenses, and Consequences," Available at http://all.net/journal/ntb/simulate/simulate.html, May 13, 1999.

[5] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," *Tech Report 79F296400*, J.P Anderson Co., Fort Washington, PA, April 15, 1980.

[6] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, February, 1987, pp. 222-232.

[7] M. Bishop, S. Cheung, and C. Wee, "The Threat from the Net," *IEEE Spectrum*, vol. 34, no. 8, August, 1997, pp. 56-63.

[8] M. Esmaili, R. Safavi-Naini, and J. Pieprzyk, "Computer Intrusion Detection: A Comparative Survey," *Tech Report 95-07*, Center for Computer Security Research, University of Wollongong, Wollongong, NSW 2522, Australia, May, 1995.

[9] A. Sundaram, "An Introduction to Intrusion Detection," *Crossroads: The ACM Student Magazine*, vol. 2, no. 4, April, 1996, pp. 26-41.

[10] H. S. Teng, K. Chen, and S. C.-Y. Lu, "Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns," in *Proc. IEEE Symp. on Research in Security and Privacy*, Oakland, CA, May 7-9, 1990, pp. 278-284.

[11] T. F. Lunt, "A Survey of Intrusion Detection Techniques," *Computers & Security*, vol. 12, no. 4, June, 1993, pp. 405-418.

[12] T. Lane, "An Application of Machine Learning to Anomaly Detection," *Tech Report 97-03*, COAST Laboratory, Department of Computer Science, Purdue University, West Lafayette, IN, February, 1997.

[13] K. Ilgun, "USTAT: A Real-Time Intrusion Detection System for UNIX," in *Proc. IEEE Symp. on Research in Security and Privacy*, Oakland, CA, May 24-26, 1993, pp. 16-28.

[14] S. Kumar and E. H. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," in *Proc. 17th National Computer Security Conf.*, Baltimore, MD, October 11-14, 1994, pp. 11-21.

[15] T. D. Garvey and T. F. Lunt, "Model Based Intrusion Detection," in *Proc. 14th National Computer Security Conf.*, Washington, DC, October 1-4, 1991, pp. 372-385.

[16] R. Anderson and A. Khattak, "The Use of Information Retrieval Techniques for Intrusion Detection," in *Proc. First International Workshop on the Recent Advances in Intrusion Detection*, Louvain-la-Neuve, Belgium, September 14-16, 1997, Available at http://www.zurich.ibm.com/pub/Other/RAID/Prog_RAID98/Talks.html#Anderson_33.

[17] C. Ko, M. Ruschitzka, and K. N. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach," in *Proc. IEEE Symposium on Security and Privacy Conf.*, Oakland, CA, May 4-7, 1997, pp. 175 - 187.

[18] R. Buschkes, M. Borning, and D. Kesdogan, "Transaction-based Anomaly Detection," in *Proc. Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, April 9-12, 1999, Available at http://ftp.sage.usenix.org/publications/library/proceedings/detection99/full_papers/buschkes/buschkes_html/index.html.

[19] E. A. Fisch, "Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior," *Ph.D. Dissertation*, Department of Computer Science, Texas A&M University, College Station, TX, 1996.

[20] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating Security Managers: A Peer-based Intrusion Detection System," *IEEE Network*, vol. 10, no. 1, January/February, 1996, pp. 20-23.

[21] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in *Proc. 20th National Information Systems Security Conf.*, Baltimore, MD, October 7-10, 1997, pp. 353-365.

[22] P. G. Neumann and P. A. Porras, "Experience with EMERALD to Date," in *Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring*,

Santa Clara, CA, April 11-12, 1999, Available at http://www2.csl.sri.com/emerald/downloads.html.

[23] F. Y. Jou, F. Gong, C. Sargor, S. F. Wu, and W. R. Cleaveland, "Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure," *Tech Report CDRL A005*, MCNC, Research Triangle Park, NC, April, 1997.

[24] K. Ilgun, "USTAT: A Real-Time Intrusion Detection System for UNIX," *M.S. Thesis*, Computer Science Department, University of California, Santa Barbara, CA, 1992.

[25] G. Vigna and R. A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System," *Computer Security*, vol. 7, no. 1, June, 1999, Available at http://www.cs.ucsb.edu/~vigna/listpub.html.

[26] G. Vigna and R. A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach," in *Proc. Computer Security Applications Conf.*, Scottsdale, AR, December 7-11, 1998, pp. 25 - 34.

[27] H. Webster, *Webster's II New Riverside University Dictionary*, Boston: The Riverside Publishing Company, 1984.

[28] M. Bishop, "A Taxonomy of UNIX System and Network Vulnerabilities," *Tech Report CSE-95-10*, Purdue University, West Lafayette, IN, May, 1995.

[29] R. Bibsey and D. Hollingworth, "Protection Analysis Project Final Report," *Tech. Report ISI/RR-78-13*, USC Information Sciences Institute, Marina del Rey, CA, May, 1978.

[30] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A Taxonomy of Computer Program Security Flaws," *ACM Computing Surveys*, vol. 26, no. 3, September, 1994, pp. 211-254.

[31] T. Aslam, "A Taxonomy of Security Faults in the Unix Operating System," *M.S. Thesis*, Department of Computer Science, Purdue University, West Lafayette, IN, 1995.

[32] R. P. Abbott, J. S. Chin, J. E. Donnelley, W. L. Konigsford, K. Tokubo, and D. A. Webb, "Security Analysis and Enhancements of Computer Operation Systems," *Tech Report NBSIR 76-1041*, National Bureau of Standards, Gaithersburg, MD, April, 1976.

[33] T. Aslam, "Use of a Taxonomy of Security Faults," *Tech Report 96-05*, COAST Laboratory, Department of Computer Science, Purdue University, West Lafayette, IN, March, 1996.

[34] M. Crosbie, B. Dole, T. Ellis, I. Krsul, and E. H. Spafford, "IDIOT - User's Guide," *Tech Report TR-96-050*, COAST Laboratory, Purdue University, West Lafayette, IN, September 4, 1996.

[35] P. G. Neumann and D. B. Parker, "A Summary of Computer Misuse Techniques," in *Proc. 13th National Computer Security Conf.*, Baltimore, MD, October 10-13, 1989, pp. 396-407.

[36] U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," in *Proc. 1997 IEEE Symp. on Security and Privacy*, Oakland, CA, May 4-7, 1997, pp. 154 - 163.

[37] S. Franklin and A. Graesser, "Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents," in *Intelligent Agents III: Proc. Workshop on Agent Theories, Architectures, and Languages,* M. J. W. a. N. R. J. Jörg P. Müller, ed., Berlin: Springer-Verlag, 1997, pp. 21-36.

[38] J. M. Bradshaw, *Software Agents*, Menlo Park, CA: AAAI Press, 1997.

[39] S. Russell and P. Norvig, "Intelligent Agents," in *Artificial Intelligence. A Modern Approach,* ed., Englewood Cliffs, NJ: Prentice-Hall, 1995, pp. 31-50.

[40] P. Maes, "Modeling Adaptive Autonomous Agents," *Artificial Life*, vol. 1, no. 1, Fall, 1994, pp. 135-162.

[41] P. Maes, "Agents That Reduce Work and Information Overload," *Communications of the ACM*, vol. 37, no. 7, July, 1994, pp. 31-41.

[42] P. Maes, "Situated Agents Can Have Goals," in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back,* P. Maes, ed., Cambridge, MA: MIT Press, 1994, pp. 49-70.

[43] M. H. Coen, "SodaBot: A Software Agent Environment and Construction System," *Tech Report AI 1493*, Massachusetts Institute of Technology, Cambridge, MA, June, 1994.

[44] H. Nwana, "Software Agents: An Overview," *Knowledge Engineering Review*, vol. 11, no. 3, November, 1996, pp. 205-244.

[45] M. R. Genesereth and S. P. Ketchpel, "Software Agents," *Communications of the ACM*, vol. 37, no. 7, July, 1994, pp. 48-53, 147.

[46] J. Yen and R. Lengari, *Fuzzy Logic: Intelligent, Control and Information*, New York: Prentice Hall, 1999.

[47] A. M. Law and W. D. Kelton, *Simulation Modeling & Analysis*, New York: McGraw-Hill, Inc., 1991.

[48] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, Upper Saddle River, NJ: Prentice-Hall, Inc., 2001.

[49] J. Humphries, "Attack Resistant Mobile Agents for Intrusion Detection Systems," *PhD Dissertation*, Department of Computer Science, Texas A&M University, College Station, TX, 2001.

[50] J. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. H. Spafford, and D. M. Zamboni, "An Architecture for Intrusion Detection Using Autonomous Agents," *Tech Report 98-05*, COAST Laboratory, Department of Computer Science, Purdue University, West Lafayette, IN, May, 1998.

[51] M. Crosbie and E. H. Spafford, "Applying Genetic Programming to Intrusion Detection," in *Proc. AAAI Fall Symp. on Genetic Programming*, Cambridge, MA, November 10-12, 1995, pp. 1-8.

[52] M. Crosbie and E. H. Spafford, "Active Defense of a Computer System Using Autonomous Agents," *Tech Report 95-008*, COAST Laboratory, Department of Computer Science, Purdue University, West Lafayette, IN, February, 1995.

[53] M. Crosbie and E. H. Spafford, "Defending a Computer System Using Autonomous Agents," *Tech Report CSD-TR-95-022*, Department of Computer Science, Purdue University, West Lafayette, IN, March, 1995.

[54] J. M. Bonifacio, E. S. Moreira, A. M. Cansian, and A. De Carvalho, "An Adaptive Intrusion Detection System Using Neural Networks," in *Proc. 14th Int. Information Security Conf. SEC'98, part of 15th IFIP World Computer Congress*, Vienna, Austria, August/September, 1998, Available at http://www.intermidia.icmc.sc.usp.br/~boni/acme/.

[55] J. M. Bonifacio, A. M. Cansian, A. De Carvalho, and E. S. Moreira, "Neural Networks Applied in Intrusion Detection Systems," in *Proc. IEEE World Congress on Computational Intelligence and Neural Networks*, Vienna, Austria, May 4-9, 1998, pp. 205 - 210.

[56] A. M. Cansian, E. S. Moreira, A. De Carvalho, and J. M. Bonifacio, "Network Intrusion Detection Using Neural Networks," in *Proc. International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'97*, Gold Coast, Australia, February, 1997, pp. 276-280.

[57] Anzen Computing, "Anzen Flight Jacket for NFR," Available at http://www.anzen.com/afj/afj_overview.html, January 24, 2000.

[58] M. Sobirey, B. Richter, and H. Konig, "The Intrusion Detection System AID," in *Proc. Intl. Conf. on Communications and Multimedia Security*, London, UK, September, 1996, pp. 278-290.

[59] M. Sobirey, "The Intrusion Detection System AID," Available at http://www-rnks.informatik.tu-cottbus.de/ sobirey/aid.e.html, January 26, 2000.

[60] J.-L. Lin, X. S. Wang, and S. J. Jajodia, "Abstraction-Based Misuse Detection: High-Level Specifications and Adaptable Strategies," in *Proc. 11th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 9-11, 1998, pp. 190 - 201.

[61] A. Mounji, B. L. Charlier, D. Zampunieris, and N. Habra, "Distributed Audit Trail Analysis," in *Proc. Internet Society Symp. on Network and Distributed System Security*, San Diego, CA, February 16-17, 1995, pp. 102 -112.

[62] A. Mounji and B. L. Charlier, "Detecting Breaches in Computer Security: A Pragmatic System with a Logic Programming Flavor," in *Proc. Eighth Benelux Workshop on Logic Programming*, Louvain-La-Neuve, Belgium, September, 1996, Available at ftp://ftp.info.fundp.ac.be/pub/users/amo/apers/benelog96.ps.Z.

[63] A. Mounji and B. L. Charlier, "Continuous Assesment of a Unix Configuration: Integrating Intrusion Detection and Configuration Analysis," in *Proc. IEEE Symp. on Network and Distributed Systems Security*, San Diego, CA, February, 1997, pp. 27 -35.

[64] G. Tsudik and R. Summers, "AudES- An Expert System for Security Auditing," *Computer Security*, vol. 6, no. 1, June, 1991, pp. 222-232.

[65] T. F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," in *Proc. 11th National Computer Security Conf.*, Baltimore, MD, October, 1988, pp. 65-73.

[66] Network Ice, "BlackIce User's Manual, version 1.0," Available at http://www.networkice.com/Support/Docs/BlackICEProUG.pdf, January 26, 2000.

[67] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," in *Proc. 7th USENIX Security Symposium*, San Antonio, TX, January, 1998, Available at http://www-nrg.ee.lbl.gov/nrg-papers.html.

[68] P. Proctor, "Audit Reduction and Misuse Detection in Heterogeneous Environments: Framework and Applications," in *Proc. 10th Annual Computer Security Applications Conf.*, Orlando, FL, December 5-9, 1994, pp. 117-125.

[69] ODS Networks, "CMDS Computer Misuse System," Available at http://www.intrusion.com/security/products/newcmds1.shtml, January 31, 2000.

[70] Network Associates, "CyberCop Intrusion Protection," Available at http://www.nai.com/international/uk/media/pdf/products/datasheet_cybercop_0699.pdf, January 24, 2000.

[71] Network Associates, "Next Generation Intrusion Detection in High Speed Networks," Available at http://www.nai.com/international/uk/media/oc/roducts/ids_nai.zip, January 24, 2000.

[72] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and an Early Prototype," in *Proc. 14th National Computer Security Conf.*, Washington, DC, October, 1991, pp. 167-176.

[73] C. Ko, D. A. Frincke, T. G. Jr., L. T. Heberlein, K. N. Levitt, B. Mukherjee, and C. Wee, "Analysis of An Algorithm for Distributed Recognition and Accountability," in *Proc. 1st ACM Conf. Computer and Communications Security*, New York, NY, November, 1993, pp. 154-164.

[74] C. Ko, G. Fink, and K. N. Levitt, "Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring," in *Proc. 10th Computer Security Applications Conf.*, Orlando, FL, December 5-9, 1994, pp. 134-144.

[75] Network Security Wizards, "Dragon Intrusion Detection," Available at http://www.network-defense.com/intro.html, January 26, 2000.

[76] L. Spitzner, "Intrusion Detection for FW-1," Available at http://www.enteract.com/~lspitz/intrusion.html, January 27, 2000.

[77] L. Mé, "GASSATA, a Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis," in *Proc. First International Workshop on the Recent Advances in Intrusion Detection*, Louvain-la-Neuve, Belgium, September 14-16, 1998, Available at http://www.zurich.ibm.com/pub/Other/RAID/rog_RAID98/table_of_content.html.

[78] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. N. Levitt, J. Rowe, S. Staniford-Chen, R. Yip, and D. Zerkle, "The Design of GrIDS: A Graph-

Based Intrusion Detection System," *Tech Report CSE-99-2*, Department of Computer Science, University of California, Davis, CA, September, 1999.

[79] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. N. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS: A Graph-based Intrusion Detection System for Large Networks," in *Proc. 19th National Information Systems Security Conf.*, Baltimore, MD, October 21-25, 1996, pp. 361 - 370.

[80] S. E. Smaha, "Haystack: An Intrusion Detection System," in *Proc. Fourth Aerospace Computer Security Applications Conference*, Austin, TX, December, 1988, pp. 37-44.

[81] D. A. Frincke, J. McConnell, D. Tobin, J. Marconi, and D. Polla, "A Framework for Cooperative Intrusion Detection," in *Proc. 21st National Information Systems Security Conf.*, Crystal City, VA, October 6-9, 1998, Available at http://csrc.nist.gov/nissc/1998/proceedings/paperF6.pdf.

[82] Y. Ho, D. A. Frincke, and D. Tobin, " Planning, Petri Nets, and Intrusion Detection," in *Proc. 21st National Information Systems Security Conf.*, Crystal City, VA, October 6-9, 1998, Available at http://csrc.nist.gov/nissc/1998/ proceedings/paperF5.pdf.

[83] H. Debar, M. Becker, and D. Siboni, "A Neural Network Component for an Intrusion Detection System," in *Proc. IEEE Computer Society Symp. on Research in Security and Privacy*, Oakland, CA, May 4-6, 1992, pp. 240-250.

[84] H. Debar and B. Dorizzi, "An Application of a Recurrent Network to an Intrusion Detection System," in *Proc. Neural Networks Conf.*, Baltimore MD, June 7-11, 1992, pp. 478 - 483.

[85] M. Asaka, S. Okazawa, and A. Taguchi, "The Implementation of IDA: An Intrusion Detection Agent System," in *Proc. 11th FIRST Conf.*, Brisbane, Australia, June, 1999, Available at http://www.ipa.go.jp/STC/IDA/index.html.

[86] M. Asaka, S. Okazawa, and A. Taguchi, "A Method of Tracing Intruders by Use of Mobile Agent," in *Proc. 9th Annual Internetworking Conf. (INET'99)*, San Jose, CA, June, 1999, Available at http://www.ipa.go.jp/STC/IDA/index.html.

[87] D. Samfat and R. Molva, "IDAMN: an Intrusion Detection Architecture for Mobile Networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, September, 1997, pp. 1373 - 1380.

[88] T. F. Lunt, "Real-Time Intrusion Detection," in *Proc. IEEE COMPCON Spring Proceedings*, Boston, MA, April, 1989, pp. 348-353.

[89] T. F. Lunt, R. Jagannathan, R. Lee, A. Whitehurst, and S. Listgarten, "Knowledge Based Intrusion Detection," in *Proc. Annual AI Systems in Government Conf.*, Washington, DC, March, 1989, pp. 102-107.

[90] S. Kumar and E. H. Spafford, "A Software Architecture to Support Misuse Intrusion Detection," *Tech Report 95-009*, Purdue University, West Lafayette, IN, March, 1995.

[91] Touch Technologies, "Intouch INSA - Network Security Agent," Available at http://www.ttisms.com/tti/nsa_www.html, January 31, 2000.

[92] Axent Technologies, "Everything You Need to Know about Intrusion Detection," *White Paper* Axent Technologies, Rockville, MD, January 31, 2000.

[93] L. T. Heberlein, B. Mukherjee, and K. N. Levitt, "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks," in *Proc. 15th National Computer Security Conf.*, Baltimore, MD, October 13-16, 1992, pp. 262-271.

[94] J. R. Winkler and J. C. Landry, "Intrusion and Anomaly Detection: ISOA Update," in *Proc. 15th National Computer Security Conf.*, Baltimore, MD, October, 1992, pp. 272-281.

[95] ODS Networks, "Kane Security Monitor," Available at http://www.intrusion.com/security/products/ksm.shtml, January 31, 2000.

[96] J. G. Hochberg, K. A. Jackson, C. A. Stallings, J. F. McClary, D. H. DuBois, and J. R. Ford, "NADIR: An Automated System for Detecting Network and File System Abuse," *Computers and Security*, vol. 12, no. 3, May, 1993, pp. 235-248.

[97] K. A. Jackson, D. H. DuBois, and C. A. Stallings, "An Expert System Application for Network Intrusion Detection," in *Proc. 14th National Computer Security Conf.*, Washington, DC, October 1-4, 1991, pp. 215-225.

[98] D. G. Simmons and R. Wilkins, "NERD: Network Event Recording Device: an Automated System for Network Anomaly Detection and Notification," in *Proc. Network and Distributed System Security Conf.*, San Diego, CA, February 16-17, 1995, pp. 87 - 93.

[99] Cisco Systems, "NetRanger 2.2.1," Available at http://www.cisco.com/univercd/cc/td/doc/product/iaabu/netrangr/nr221/nr221ug/index.htm, January 31, 2000.

[100] D. Anderson, T. Frivold, and A. Valdes, "Next Generation Intrusion Detection Expert System (NIDES): A Summary," *Tech Report SRI-CSL-95-07*, SRI International, Menlo Park, CA, May, 1995.

[101] D. S. Bauer and M. E. Koblentz, "NIDX - A Real-Time Intrusion Detection Expert System," in *Proc. USENIX 1988 Technical Conf.*, San Francisco, CA, Summer, 1988, pp. 261-273.

[102] L. T. Heberlein, K. N. Levitt, and B. Mukherjee, "A Method to Detect Intrusive Activity in a Networked Environment," in *Proc. 14th National Computer Security Conf.*, Washington, DC, October, 1991, pp. 362-371.

[103] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor," in *Proc. IEEE Symp. on Research in Security and Privacy*, Oakland, CA, May 7-9, 1990, pp. 296-304.

[104] PRC Incorporated, "PRC PreCis," Available at http://www.bellevue.prc.com/precis/solution.pdf, February 1, 2000.

[105] Internet Security Systems, "RealSecure Data Sheet," Available at http://solutions.iss.net/products/rsecure/realsecure.pdf, February 2, 2000.

[106] L3 Network Security, "Retriever 1.5 FAQ," Available at http://www.l-3security.com/Products/Retriever/, February 3, 2000.

[107] D. M. Zamboni, "SAINT: A Security Analysis Integration Tool," in *Proc. Systems Administration, Networking and Security Conf.*, Washington DC, May 12-18, 1996, Available at ftp://coast.cs.purdue.edu/pub/doc/tools/SAINT.ps.gz.

[108] MineStar Inc, "SecureNet Pro Frequently Asked Questions," Available at http://www.mimestar.com/html/product_faq.htm, February 2, 2000.

[109] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proc. USENIX LISA '99 - 13th Systems Administration Conf.*, Seattle, WA, November 10-13, 1999, Available at http://www.clark.net/~roesch/lisapaper.txt.

[110] P. A. Porras, "STAT A State Transition Analysis Tool For Intrusion Detection," *M.S. Thesis*, Department of Computer Science, University of California, Santa Barbara, CA, 1992.

[111] S. E. Hansen and E. T. Atkins, "Automated System Monitoring and Notification with SWATCH," in *Proc. USENIX Systems Administration Conf. (LISA VII)*, Monterey, CA, November, 1993, pp. 145-155.

[112] En Garde Systems, "T-Sight on Target Security," Available at http://www.engarde.com/software/t-sight/index.html, February 7, 2000.

[113] G. G. Christoph, K. A. Jackson, M. C. Neuman, C. L. B. Siciliano, D. D. Simmonds, C. A. Stallings, and J. L. Thompson, "UNICORN: Misuse Detection

for UNICOS," in *Proc. SuperComputing Conf.*, San Diego, CA, December 3-5, 1995, CD-ROM only.

[114] G. E. Liepins and H. S. Vaccaro, "Anomaly Detection: Purpose and Framework," in *Proc. 12th National Computer Security Conf.*, Washington, D.C., October, 1989, pp. 495-504.

[115] H. S. Vaccaro and G. E. Liepins, "Detection of Anomalous Computer Session Activity," in *Proc. IEEE Symp. on Research in Security and Privacy*, Oakland, CA, May 1-3, 1989, pp. 280-289.

APPENDIX A

SURVEYED SYSTEMS

System	Reports	Alarms	Manual Intrusion Response	Automatic Intrusion Response	References
AAFID	✓				[50-53]
ACME!	✓				[54-56]
AFJ	✓	✓		✓	[57]
AID		✓			[58, 59]
ARMD	✓		✓		[60]
ASAX	✓				[61-63]
AudES	✓				[64]
Audit	✓		✓		[65]
BlackICE	✓	✓		✓	[66]
Bro	✓	✓			[67]
CMDS1	✓	✓			[68]
CMDS2	✓	✓			[69]
CSM	✓	✓	✓	✓	[19, 20]
CyberCop	✓	✓			[70, 71]
DIDS	✓		✓		[72]
Discovery	✓				[65]
DPEM		✓			[17, 73, 74]
Dragon	✓				[75]
Emerald	✓			✓	[21, 22]
FW-1		✓			[76]
GASSATA	✓				[77]
GrIDS		✓			[78, 79]
Haystack	✓				[80]
Hummingbird	✓		✓		[81, 82]
Hyperview		✓			[83, 84]
IDA	✓			✓	[85, 86]
IDAMN		✓	✓		[87]
IDES	✓	✓			[65, 88, 89]
IDIOT	✓			✓	[14, 34, 90]

System	Reports	Alarms	Manual Intrusion Response	Automatic Intrusion Response	References
INSA	✓	✓	✓	✓	[91]
Intruder Alert	✓	✓		✓	[92]
ISM	✓				[93]
ISOA	✓		✓		[94]
JiNao	✓	✓		✓	[23]
Kane Security Monitor	✓	✓			[95]
MIDAS		✓			[65]
NADIR	✓		✓		[96, 97]
NERD		✓			[98]
NetRanger	✓	✓	✓	✓	[99]
NetSTAT	✓	✓		✓	[25, 26]
NIDES	✓	✓			[100]
NIDX	✓	✓		✓	[101]
NSM	✓	✓			[102, 103]
PréCis	✓	✓			[104]
RealSecure	✓	✓		✓	[105]
Retriever	✓	✓			[106]
SAINT	✓				[107]
SecureNet Pro	✓	✓		✓	[108]
Snort	✓	✓			[109]
STAT		✓		✓	[110]
SWATCH		✓		✓	[111]
TIM		✓			[10]
T-Sight	✓	✓	✓		[112]
UNICORN	✓	✓			[113]
USTAT	✓	✓		✓	[13, 24]
Wisdom and Sense	✓				[114, 115]

APPENDIX B

MAPPING BETWEEN PLAN STEPS AND TACTICS

Gather Evidence	Preserve Evidence
Enable additional logging Enable process accounting Enable additional IDS Trace the connection Employ honey-pot Employ smoke-pot Contact the servicing ISP	Enable remote logging Enable logging to unchangeable media Enable additional IDS Restrict user activity Create backups
Communicate with Attacker	Social Engineering
Warn the intruder Force additional authentication	Warn the intruder Force additional authentication
Slow the Attack	Stop the Attack
Employ honey-pot Employ smoke-pot Force additional authentication Restrict user activity Turn off modems Lock user account Suspend user jobs Terminate user session Block IP address Disable the attacked ports or services Disconnect from the network	Turn off modems Lock user account Suspend user jobs Terminate user session Block IP address Disable the attacked ports or services Shutdown host Disconnect from the network
Identify Damaged Files	Protect Critical Files
Enable additional logging Enable process accounting Enable additional IDS	Create backups Employ temporary shadow files
Notify the System Administrator	Attack the attacking system
Generate a report Generate an alarm	Denial of service attack System compromise attack

APPENDIX C

TACTICS AND IMPLEMENTATIONS

Enable additional logging	Enable remote logging
Enable additional logging	Enable remote logging - machine Gabriel Enable remote logging - machine Limbo
Enable logging to unchangeable media	Enable additional IDS
Enable logging to Printer Enable logging to CD-ROM	Enable an additional IDS - Black Enable an additional IDS - Gold
Enable process accounting	Contact the ISP
Enable process accounting	Contact the ISP by Email
Trace the connection	Warn the intruder
Reverse DNS Lookup Agent-based approach	Warn the intruder -Email Warn the intruder -Talk
Employ honey-pot	Employ smoke-pot
Employ honey-pot 1 Employ honey-pot 2	Employ smoke-pot 1 Employ smoke-pot 2
Force additional authentication	Block IP address
User name and password again Ask secret phrase	Block IP address - At the host Block IP address - At the router
Lock user account	Suspend user jobs
Lock user account	Suspend user jobs
Restrict user activity	Turn off modems
Restrict user activity	Turn off modems
Terminate user session	Shutdown host
Terminate user session	Shutdown host
Disable the attacked ports or services	Create backups
All Ports Only the attacked port	Complete system Critical system files
Disconnect from the network	Employ temporary shadow files
Disconnect from the network	Employ temporary shadow files
Generate a report	Generate an alarm
Generate a report	Generate an alarm - email Generate an alarm - pager Generate an alarm - speaker announcement
Denial of service attack	System compromise attack
Denial of service attack SMURF Denial of service attack Fraggle Denial of service attack Tribe Flood	System Compromise Attack automountd System Compromise Attack ping System Compromise Attack GetAdmin

APPENDIX D

RESPONSE GOAL CLASSIFICATION MATRIX

The response goal classification table lists the effect of the response goal classification on the formation of a tentative plan. The response goal is an evaluative criterion. Cells contain a value between 0.20 and 1.00 representing the importance of a PTI in obtaining a particular response goal. Notify system administrator is important regardless of the response goal. Higher values indicate more appropriate PTI.

The reasoning employed in the formulation of this table is as follows.

- Analyze Attack: The goal is to gather as much information as possible during the attack so that it can be analyzed. Those plan steps that facilitate this goal (gather evidence, preserve evidence, communicate with the attacker, social engineering) are heavily weighted. Identifying damaged files is weighted less but still heavily weighted. Attacking backing and slowing the attack are discouraged while stopping the attack receives the lowest possible weight.
- Catch Attack: The goal is to identify and catch the attacker so that future action can be taken against the attacker. Those plan steps that facilitate this goal (gather evidence and preserve evidence) are heavily weighted. Slowing the attack, identifying damaged files, and social engineering are weighted less but still heavily weighted. Communicating with the attacker, stopping the attack, protecting critical files, and attacking back are discouraged.

- Mask Attack: The goal is mask the attack so that service is not disrupted and the attack is terminated as soon as possible. Those plan steps that facilitate this goal (stop the attack, protect critical files, and attack back) are heavily weighted. Gather evidence, communicate with the attacker, slow the attack, identify damaged files, and social engineering are weighted less but still heavily weighted. Preserve evidence is discouraged.
- Maximize Confidentially: The goal is to prevent the disclosure of information. Those plan steps that facilitate this goal (stop the attack, identify damaged files, and attack back) are heavily weighted. Gather evidence, preserve evidence, communicate with the attacker, slow the attack, protect critical files, and social engineering are weighted less but still are heavily weighted.
- Maximize Data Integrity: The goal is to prevent the changing of files on the system so that their integrity is maintained. Those plan steps that facilitate this goal (gather evidence, preserve evidence, stop the attack, protect critical files, and attack back) are heavily weighted. Communicate with the attacker, slow the attack, and social engineering are weighted less but are still heavily weighted. Identify damaged files is discouraged.
- Minimize Cost: The goal is to minimize the cost of implementing a response in terms of resources. This goal would be appropriate on a non-critical system that is routinely rebuilt. Those plan steps that facilitate this

goal (stop the attack, protect critical system files, attack back and social engineering) are heavily weighted with attacking back being the highest weighted response for this goal. All other plan steps are discouraged.

- Recover Gracefully: The goal is not to stop the intrusion but to retain the capability to recover gracefully from any attack with minimal effort. Those plan steps that facilitate this goal (gather evidence, preserve evidence, and protect critical files) are heavily weighted. Communicate with the attacker, slow the attack, stop the attack, and social engineering is weighted less but still heavily weighted. Attacking back is discouraged.
- Sustain Service: The goal is sustain service during any attack. While this goal is similar to mask the attack, it is differentiated in that there is no attempt to stop the attack - providing service is the paramount goal. Those plan steps that facilitate this goal (slow the attack, protect critical files, and attack back) are heavily weighted. Gather evidence, communicate with the attacker, identify damaged files, and social engineering are weighted less but still heavily weighted. Preserve evidence and stopping the attack are discouraged.

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Plan Steps								
Gather Evidence	.8	.8	.6	.8	.8	.4	.8	.6
Preserve Evidence	.8	.8	.4	.8	.8	.4	.8	.4
Communicate with attacker	.8	.4	.6	.6	.6	.4	.6	.6
Slow the attack	.4	.6	.6	.6	.6	.4	.6	.8
Stop the attack	.2	.4	.8	.8	.8	.8	.6	.4
Identify damaged files.	.6	.6	.6	.8	.4	.4	1	.6
Protect critical files	.4	.4	.8	.6	.8	.8	.8	.8
Notify System Administrator	1	1	1	1	1	1	1	1
Attack back	.4	.4	.8	.8	.8	1	.4	.8
Social Engineering	.8	.6	.6	.6	.6	.8	.6	.6
Tactics								
Additional logging	1	1	.2	.2	.8	.4	.8	.4
Remote logging	1	1	.2	.2	.8	.4	.6	.4
Unchangeable logging	1	1	.2	.2	1	.2	.6	.4
Process accounting	.8	.8	.2	.2	.6	.4	.6	.6

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Additional IDS	1	1	.2	.2	.8	.4	.6	.6
Trace the connection	1	1	.4	.2	.2	.4	.4	.4
Honey-pot	1	.8	.8	.8	.8	.4	.6	.8
Smoke-pot	.8	1	.8	.8	.8	.4	.6	.8
Contact the ISP	.8	.8	.6	.8	.8	.8	.6	.4
Warn intruder	.8	.6	.6	.6	.6	.8	.6	.6
Additional authentication	.6	.6	.6	.8	.8	.6	.6	.8
Restrict user activity	.4	.4	.8	.6	.6	.8	.8	.8
Turn off modems	.2	.2	.2	1	1	.4	.4	.2
Lock user account	.4	.2	1	1	1	.8	.8	.6
Suspend user jobs	.4	.4	.8	.6	.6	.8	.8	.8
Terminate user session	.4	.4	1	.8	.8	.8	.8	.4
Block IP address	.2	.4	.8	.8	.6	.6	.6	.2
Disable attacked ports	.4	.2	.4	.8	.8	.8	.6	.2
Shutdown host	.2	.2	.2	1	1	.4	.2	.2
Disconnect from network	.2	.2	.2	1	1	.4	.4	.2
Create backups	.8	.6	.6	.2	1	.4	1	.6

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Temporary shadow files	.6	.6	.8	.2	1	.6	.8	1
Generate report	1	1	1	1	1	1	1	1
Generate alarm	1	1	1	1	1	1	1	1
Denial of service attack	.4	.4	.8	.8	.8	1	.4	.8
System compromise attack	.4	.4	.8	.8	.8	.8	.6	.6
Implementations								
Additional logging	1	1	.2	.2	.8	.4	.8	.4
Block IP at host	.4	.4	.8	.8	.6	.6	.6	.4
Block IP at router	.2	.2	1	.8	.6	.6	.6	.2
Contact ISP by Email	.8	.8	.6	.8	.8	.8	.6	.4
Complete system backup	.8	.8	.4	.2	1	.2	1	.4
Critical system files backups	.8	.6	.6	.2	.8	.4	.8	.6
DOS attack SMURF	.4	.4	.8	.8	.8	1	.4	.8
DOS attack Fraggle	.4	.4	.8	.8	.8	1	.4	.8
DOS attack Tribe Flood	.4	.4	.8	.8	.8	1	.4	.8

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Disable all ports	.4	.2	.2	1	.8	.6	.6	.2
Disable attacked ports	.4	.4	.4	.8	.8	.8	.6	.2
Disconnect from network	.2	.2	.2	1	1	.4	.4	.2
Temporary Shadow files	.6	.6	.8	.2	1	.6	.8	1
Honey-pot 1	1	.8	.8	.8	.8	.4	.6	.8
Honey-pot 2	1	.8	.8	.8	.8	.4	.6	.8
Smoke-pot 1	.8	1	.8	.8	.8	.4	.6	.8
Smoke-pot 2	.8	1	.8	.8	.8	.4	.6	.8
IDS Black	1	1	.2	.2	.8	.4	.6	.6
IDS Gold	1	1	.2	.2	.8	.4	.6	.6
Ask user name and password	.6	.6	.6	.8	.8	.6	.6	.8
Ask secret phrase	.6	.6	.6	.8	.8	.6	.6	.8
Incident report	1	1	1	1	1	1	1	1
Alarm - email	.6	.2	.8	1	1	1	1	1
Alarm - pager	1	1	1	1	1	1	1	1
Alarm - speaker announcement	1	1	.4	1	1	1	1	1
Lock user account	.4	.2	1	1	1	.8	.8	.2
Logging to printer	1	1	.2	.2	1	.2	.6	.4

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Logging to CD-ROM	1	1	.2	.2	1	.4	.6	.4
Process accounting	.8	.8	.2	.2	.6	.4	.6	.6
Remote logging to Gabriel	1	1	.2	.2	.8	.4	.6	.4
Remote logging to Limbo	1	1	.2	.2	.8	.4	.6	.4
Restrict user activity	.4	.4	.8	.6	.6	.8	.8	.8
Shutdown host	.2	.2	.2	1	1	.4	.2	.2
Suspend user jobs	.4	.4	.8	.6	.6	.8	.8	.8
Counterattack automountd	.4	.4	.8	.8	.8	.8	.8	.8
Counterattack ping	.4	.4	.8	.8	.8	.8	.8	.8
Counterattack GetAdmin	.4	.4	.8	.8	.8	.8	.8	.8
Terminate user session	.4	.4	1	.8	.8	.8	.8	.4
Reverse DNS Lookup	1	1	.4	.2	.2	.4	.4	.4
Agent-based approach	1	1	.4	.2	.2	.4	.4	.4
Turn off modems	.4	.4	.2	.8	.8	.4	.4	.2

PTI	Analyze Attack	Catch Attack	Mask Attack	Maximize Confidentially	Maximize Data Integrity	Minimize Cost	Recover Gracefully	Sustain Service
Warn the intruder -Email	.8	.6	.6	.6	.6	.8	.6	.6
Warn the intruder -Talk	.8	.8	.6	.8	.8	1	.6	.8

APPENDIX E

SUSPICION MATRIX

The suspicion matrix table lists the effect of suspicion on the formation of a tentative plan. Suspicion is a decision criterion and darkened cells are constrained so that constrained PTI cannot be a component of an implemented plan.

The reasoning employed in the formulation of this table is as follows:

- Plan Step level: All plan steps are viable under all suspicion levels except for attacking back and social engineering. Attacking back has significant implications associated with it. As such, it is only viable when there is a very high level of belief that the system is under attack. Similarly, social engineering attempts to manipulate the user into taking certain actions that likewise have substantial implications although not as significant as attacking back. As such, it is constrained but not as severely as attacking back.
- Tactics level: Tactics supporting the plan steps attack back and social engineering are constrained at the same level as their parent plan step. Additionally, those tactics that have system-wide implications are constrained. Shutting down the host and disconnecting from the network are the most severe tactics and are reserved for only those incidents where there is a very high degree of suspicion. Disabling ports and turning off modems are less severe tactics but have system-wide implications as they

can lead to the shutdown of services. Finally, blocking IP addresses has the potential of denying service to valid users in a shared terminal environment. It is constrained so that there must be an appropriate degree of suspicion that the system is under attack before this response becomes viable.

- Implementation level: Implementations supporting the plan steps attack back and social engineering are constrained at the same level as their parent plan step. Implementations supporting the tactics shutdown host, disconnect from the network, turn off modems, disable attacked ports, and block IP addresses are constrained at the same level as their parent tactic. Additionally, the implementations create complete system backups and lock user accounts are constrained at a low level. Creating complete system backups is a resource intensive task that should not be implemented when there is limited suspicion of an attack. Lock user accounts is constrained because there are other implementations such as terminate a user session or suspend user jobs that are more appropriate at very low levels of suspicion.

PTI	Suspicion Level			
	.25	.5	.75	1.0
Plan Step				
Gather Evidence				
Preserve Evidence				
Communicate with the attacker				
Slow the attack				
Stop the attack				
Identify potential damaged files.				
Protect critical system files				
Notify the System Administrator				
Attack the attacking system	■	■	■	
Social Engineering	■	■		
Tactics				
Enable additional logging				
Enable remote logging				
Enable logging to unchangeable media				
Enable process accounting				
Enable additional IDS				
Trace the connection				
Employ honey-pot				
Employ smoke-pot				
Contact the ISP	■	■		
Warn the intruder				
Force additional authentication				
Restrict user activity				
Turn off modems	■	■		
Lock user account				
Suspend user jobs				
Terminate user session				
Block IP address	■			
Disable the attacked ports or services	■	■		
Shutdown host	■	■	■	
Disconnect from the network	■	■	■	
Create backups				
Employ temporary shadow files				
Generate a report				
Generate an alarm				
Denial of service attack	■	■	■	
System compromise attack	■	■	■	
Implementations				
Enable additional logging implementation				

PTI	Suspicion Level			
	.25	.5	.75	1.0
Block IP address - At the host				
Block IP address - At the router	■			
Contact the ISP by Email	■	■		
Create backups - Complete system	■			
Create backups - Critical system files				
Denial of service attack SMURF	■	■	■	
Denial of service attack Fraggle	■	■	■	
Denial of service attack Tribe Flood	■	■	■	
Disable the attacked ports or services - All	■	■		
Disable the attacked ports - Only the attacker	■	■		
Disconnect from the network implementation	■	■	■	
Employ temporary shadow files implementation				
Employ honey-pot 1				
Employ honey-pot 2				
Employ smoke-pot 1				
Employ smoke-pot 2				
Enable an additional IDS - Black				
Enable an additional IDS - Gold				
Force additional authentication - user name/password				
Force additional authentication - secret phrase				
Generate an incident report				
Generate an alarm - email				
Generate an alarm - pager				
Generate an alarm - speaker announcement				
Lock user account implementation	■			
Enable logging to unchangeable media - Printer				
Enable logging to unchangeable media - CD-ROM				
Enable process accounting implementation				
Enable remote logging - machine Gabriel				
Enable remote logging - machine Limbo				
Restrict user activity implementation				
Shutdown host implementation	■	■	■	
Suspend user jobs implementation				

PTI	Suspicion Level			
	.25	.5	.75	1.0
System Compromise Attack automountd	■	■	■	
System Compromise Attack ping	■	■	■	
System Compromise Attack GetAdmin	■	■	■	
Terminate user session implementation				
Trace the connection -Reverse DNS Lookup				
Trace the connection -Agent-based approach				
Turn off modems implementation	■	■		
Warn the intruder -Email				
Warn the intruder -Talk				

APPENDIX F

TIME OF ATTACK MATRIX

The time of attack metric table lists the effect of time of attack on the formation of a tentative plan. Time of attack is a decision criterion and darkened cells are constrained so that constrained PTI cannot be a component of an implemented plan.

The reasoning employed in the formulation of this table is as follows.

- Plan Step level: Slowing or stopping the attack is not viable after the attack has concluded.
- Tactics level: Similarly, forcing authentication, tracing a connection, suspending user jobs, terminating a user connection, and employing shadow files are only viable when the user is actively connected to the system.
- Implementations level: Implementations supporting the tactics force authentication, trace the connection, suspend user jobs, terminate the connection, and employ shadow files are constrained similarly to their parent tactic. Creating a complete system backup is not viable when the system is under active attack in all but the most contrived cases. Finally, warning the intruder through talk is only appropriate when the intruder is logged onto the system.

PTI	Before	During	After
Plan Steps			
Gather Evidence			
Preserve Evidence			
Communicate with the attacker			
Slow the attack			■
Stop the attack			■
Identify potential damaged files.			
Protect critical system files			
Notify the System Administrator			
Attack the attacking system			
Social Engineering			
Tactics			
Enable additional logging			
Enable remote logging			
Enable logging to unchangeable media			
Enable process accounting			
Enable additional IDS			
Trace the connection			■
Employ honey-pot			
Employ smoke-pot			
Contact the ISP			
Warn the intruder			
Force additional authentication			■
Restrict user activity			
Turn off modems			
Lock user account			
Suspend user jobs			■
Terminate user session			■
Block IP address			
Disable the attacked ports or services			
Shutdown host			
Disconnect from the network			
Create backups			
Employ temporary shadow files			■
Generate a report			
Generate an alarm			
Denial of service attack			
System compromise attack			
Implementations			
Enable additional logging implementation			
Block IP address - At the host			

PTI	Before	During	After
Block IP address - At the router			
Contact the ISP by Email			
Create backups - Complete system		■	
Create backups - Critical system files			
Denial of service attack SMURF			
Denial of service attack Fraggle			
Denial of service attack Tribe Flood			
Disable all ports or services			
Disable attacker's ports or service			
Disconnect from the network implementation			
Employ temporary shadow files implementation			■
Employ honey-pot 1			
Employ honey-pot 2			
Employ smoke-pot 1			
Employ smoke-pot 2			
Enable an additional IDS - Black			
Enable an additional IDS - Gold			
Force additional authentication user name/password			■
Force additional authentication - secret phrase			■
Generate an incident report			
Generate an alarm - email			
Generate an alarm - pager			
Generate an alarm - speaker announcement			
Lock user account implementation			
Enable logging to unchangeable media - Printer			
Enable logging to unchangeable media - CD-ROM			
Enable process accounting implementation			
Enable remote logging - machine Gabriel			
Enable remote logging - machine Limbo			
Restrict user activity implementation			
Shutdown host implementation			
Suspend user jobs implementation			■
System Compromise Attack automountd			
System Compromise Attack ping			
System Compromise Attack GetAdmin			
Terminate user session implementation			■
Trace the connection -Reverse DNS Lookup			■
Trace the connection -Agent-based approach			■
Turn off modems implementation			
Warn the intruder -Email			

PTI	Before	During	After
Warn the intruder -Talk			■

APPENDIX G

TYPE OF ATTACKER CLASSIFICATION MATRIX

The type of attacker classification table lists the effect of the attacker classification on the formation of a tentative plan. Type of attacker is an evaluative criterion. Cells contain a value between 0.25 and 1.00 representing the importance of a PTI against a particular type of attacker. Higher values indicate more appropriate PTI. The reasoning employed in the formulation of this table is as follows.

- Plan Step level:
  - The plan steps of gather evidence, identify potential damaged files, protect critical system files, and notify system administrator are not affected by the type of attacker. These plan steps are annotated with a dash.
  - If an expert is attacking, preserving evidence becomes more important as an expert will take more sophisticated steps to cover his tracks. If a novice is attacking, preserving evidence becomes less important as novices are easier to detect and will not be as successful in altering system logs of their activity.
  - If a human is attacking, social engineering and communicating with the attacker become appropriate. If an automated program is attacking, social engineering and communicating with the attacker becomes less important as the program is unlikely to respond to social engineering attempts. If a

novice is attacking, social engineering and communicating with the attacker becomes more important as novices can be more easily scared or manipulated. More experienced attackers may not be so easily manipulated. Fundamental to the attacker is the idea of being anonymous - communicating and social engineering attacks remove this defense to a certain extent.

  - Slow the attack and stop the attack are related. A novice can be more easily stopped and this is especially true when the attack is not automated. An expert is more difficult to stop and slowing the attack, as opposed to stopping the attack, becomes a more appropriate plan step.
  - Attacking back is going to be more successful against novices as they have fewer defenses against attacks in general. Likewise, attacking back is going to be more effective in the short term if the attack is human-based and thus the human is present to notice that he is actively being attacked back.

- Tactic and implementation levels: The logic employed at these levels is consistent with the logic articulated for the plan step level.

PTI	Novice Automated	Novice Human	Expert Automated	Expert Human
Plan Step				
Gather Evidence	-	-	-	-
Preserve Evidence	.5	.5	.75	.75
Communicate with the attacker	.5	1	.25	.75
Slow the attack	.5	.25	.75	.75
Stop the attack	.75	.75	.5	.5
Identify potential damaged files	-	-	-	-
Protect critical system files	-	-	-	-
Notify the System Administrator	-	-	-	-
Attack the attacking system	1	.75	.25	.5
Social Engineering	.5	1	.25	.75
Tactics				
Enable additional logging	-	-	-	-
Enable remote logging	.5	.5	1	1
Enable logging to unchangeable media	.5	.25	1	1
Enable process accounting	1	1	.5	.25
Enable additional IDS	1	1	1	.75
Trace the connection	1	1	.75	.75
Employ honey-pot	.25	1	.5	.75
Employ smoke-pot	.25	1	.5	.75
Contact the ISP	.5	.25	1	1
Warn the intruder	.25	1	.25	.5
Force additional authentication	1	1	1	.75
Restrict user activity	1	1	.5	.25
Turn off modems	1	1	1	.75
Lock user account	1	1	1	.5
Suspend user jobs	1	1	.5	.25
Terminate user session	1	1	1	.5
Block IP address	1	.75	.75	.5
Disable the attacked ports or services	1	.75	.75	.5
Shutdown host	-	-	-	-
Disconnect from the network	-	-	-	-
Create backups	.5	1	.25	.5
Employ temporary shadow files	1	1	1	.75
Generate a report	-	-	-	-
Generate an alarm	.75	.75	1	1
Denial of service attack	.75	1	.5	.5
System compromise attack	1	1	.75	.5

PTI	Novice Automated	Novice Human	Expert Automated	Expert Human
Implementations				
Enable additional logging implementation	-	-	-	-
Block IP address - At the host	1	.75	.75	.5
Block IP address - At the router	1	1	1	.75
Contact the ISP by Email	.25	.5	.5	1
Create backups - Complete system	.25	1	.25	1
Create backups - Critical system files	-	-	-	-
Denial of service attack SMURF	.75	1	.5	.5
Denial of service attack Fraggle	.75	1	.5	.5
Denial of service attack Tribe Flood	.75	1	.5	.5
Disable the attacked ports or services - All	-	-	-	-
Disable the attacked ports or services - Only the attacker	1	.75	.75	.5
Disconnect from the network implementation	-	-	-	-
Employ temporary shadow files implementation	1	1	1	.75
Employ honey-pot 1	.25	1	.5	.75
Employ honey-pot 2	.25	1	.5	.75
Employ smoke-pot 1	.25	1	.5	.75
Employ smoke-pot 2	.25	1	.5	.75
Enable an additional IDS - Black	1	1	1	.75
Enable an additional IDS - Gold	1	1	1	.75
Force additional authentication - ask user name and password	1	1	1	.75
Force additional authentication - ask secret phrase	1	1	1	.75
Generate an incident report	-	-	-	-
Generate an alarm - email	.5	1	.25	.25
Generate an alarm - pager	.75	.5	1	1
Generate an alarm - speaker announcement	.75	.5	1	1
Lock user account implementation	1	1	1	.5
Enable logging to unchangeable media - Printer	.5	.25	1	1
Enable logging to unchangeable media - CD-ROM	.5	.25	1	1

PTI	Novice Automated	Novice Human	Expert Automated	Expert Human
Enable process accounting implementation	1	1	.5	.25
Enable remote logging - machine Gabriel	.5	.5	1	1
Enable remote logging - machine Limbo	.5	.5	1	1
Restrict user activity implementation	1	1	.5	.25
Shutdown host implementation	-	-	-	-
Suspend user jobs implementation	1	1	.5	.25
System Compromise Attack automountd	1	1	.75	.5
System Compromise Attack ping	1	1	.75	.5
System Compromise Attack GetAdmin	1	1	.75	.5
Terminate user session implementation	1	1	.75	.25
Trace the connection -Reverse DNS Lookup	1	1	.75	.5
Trace the connection -Agent-based approach	.75	.75	1	.75
Turn off modems implementation	1	1	1	.75
Warn the intruder -Email	.75	.75	.75	.75
Warn the intruder -Talk	.25	1	.25	.75

APPENDIX H

TYPE OF ATTACK CLASSIFICATION MATRIX

The response goal classification table lists the effect of the response goal classification on the formation of a tentative plan. The system response goal is set by the system administrator and is an evaluative criterion. Cells contain a value between 0.25 and 1.00 representing the importance of a PTI against a particular response goal. Higher values indicate more appropriate PTI. The reasoning employed in the formulation of this table is as follows.

- Some responses should be executed regardless of the type of attack. Generate a report is an example.
- In weighting responses, the severity of the attack and the cost of implementing the response were dominant factors.
- Certain responses only affect the attacker and not the rest of the users on the system (e.g. lock user account). These responses were always implemented regardless of the type of the attack. Responses that affected all of the users of a system were limited to the most severe forms of attack.
- Certain responses such as suspending user jobs were considered appropriate for less severe attacks but inappropriate for more severe attacks where a more drastic response was deemed appropriate.

PTI	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)
Plan Step														
Gather Evidence	.75	1	.75	.75	1	1	.5	.5	.5	.5	.75	.75	1	1
Preserve Evid	.5	1	.5	.5	1	1	.25	.25	.25	.25	.75	.75	1	1
Com with attker	.75	.75	.75	.75	.75	.75	1	.75	.75	.5	.5	.5	.5	.5
Slow the attack	.75	.75	.75	.75	.5	.5	1	1	.75	.75	.75	.75	.75	.75
Stop the attack	.75	.75	.75	1	1	1	1	1	1	1	1	1	1	1
Id pot dmg files.	1	1	.5	.75	1	1	.25	.25	.25	.25	.75	.75	1	.25
Prot crit sys files	.5	.5	.5	.75	1	1	.25	.25	.25	.25	.5	.75	1	.25
Notify SA	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Attack back	.25	.5	.25	.5	.75	1	.25	.5	1	1	.25	.5	1	1
Social Eng	.75	.75	1	.75	.5	.5	1	.5	.75	.5	.5	.5	.5	.5
Tactics														
Enable additional logging	1	1	1	1	1	1	.25	.25	.25	.25	1	1	1	1
Enable remote logging	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Enable logging to unchangeable media	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Enable process accounting	.75	1	.75	.75	1	1	.25	.25	.25	.25	1	1	1	1
Enable additional IDS	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Trace the connection	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Employ honey-pot	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Employ smoke-pot	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Contact the ISP	.25	1	.25	.5	1	1	1	1	1	1	.25	.5	1	1
Warn the intruder	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Force additional authentication	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Restrict user activity	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Turn off modems	.25	.5	.25	.75	1	1	.25	.5	1	1	.25	.5	1	1

PTI	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)
Lock user account	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Suspend user jobs	1	1	1	1	.5	.25	.75	.5	.25	.25	.75	.5	.25	.25
Terminate user session	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Block IP address	.25	.5	.25	.75	1	1	.25	.5	1	1	.25	.75	1	1
Disable the attacked ports or services	.25	.5	.25	.75	1	1	.25	.5	1	1	.25	.75	1	1
Shutdown host	.25	.5	.25	.5	.5	.75	.25	.25	.75	1	.25	.5	1	1
Disconnect from the network	.25	.5	.25	.5	.5	.75	.25	.25	.75	1	.25	.5	1	1
Create backups	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	1	1	1	1
Employ temporary shadow files	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	1	1	1	1
Generate a report	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Generate an alarm	.75	1	.75	1	1	1	1	1	1	1	.75	1	1	1
Denial of service attack	.25	.5	.25	.5	.75	1	.25	.25	.25	.25	.25	.5	1	1
System compromise attack	.25	.5	.25	.5	.75	1	.25	.75	1	1	.25	.5	1	1
Implementations														
Enable additional logging implementation	1	1	1	1	1	1	.5	.5	.5	.5	1	1	1	1
Block IP address - At the host	.25	.5	.25	.75	1	1	1	1	1	1	.25	.75	1	1
Block IP address - At the router	.25	.5	.25	.75	1	1	1	1	1	1	.25	.75	1	1
Contact the ISP by Email	.25	1	.25	.5	1	1	1	1	1	1	.25	.5	1	1
Create backups - Complete system	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	1	.25
Create backups - Critical system files	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	1	1	.5	1
Denial of service attack SMURF	.25	.5	.25	.5	.75	1	.25	.25	.25	.25	.25	.5	1	1

PTI	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)
Denial of service attack Fraggle	.25	.5	.25	.5	.75	1	.25	.25	.25	.25	.25	.5	1	1
Denial of service attack Tribe Flood	.25	.5	.25	.5	.75	1	.25	.25	.25	.25	.25	.5	1	1
Disable the attacked ports or services - All	.25	.25	.25	.25	.5	.75	.25	.25	.75	1.0	.25	.5	1	1
Disable the attacked ports or services - Only the attacker	.25	.5	.25	.75	1	1	.25	.5	1	1	.25	.75	1	1
Disconnect from the network implementation	.25	.5	.25	.5	.5	.75	.25	.25	.75	1	.25	.5	1	1
Employ temporary shadow files implementation	.25	.25	.25	.25	.25	.25	.25	.25	.25	.25	1	1	1	1
Employ honey-pot 1	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Employ honey-pot 2	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Employ smoke-pot 1	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Employ smoke-pot 2	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Enable an additional IDS - Black	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Enable an additional IDS - Gold	.5	1	.5	1	1	1	1	1	1	1	.5	1	1	1
Force additional authentication - ask user name and password	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Force additional authentication - ask secret phrase	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Generate an incident report	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Generate an alarm - email	1	.75	1	.75	.75	.75	.75	.75	.75	.75	1	.75	.75	.75
Generate an alarm - pager	.75	1	.75	1	1	1	1	1	1	1	.75	1	1	1

PTI	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)
Generate an alarm - speaker announcement	.75	1	.75	1	1	1	1	1	1	1	.75	1	1	1
Lock user account implementation	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Enable logging to unchangeable media - Printer	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Enable logging to unchangeable media - CD-ROM	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Enable process accounting implementation	.75	1	.75	1	1	1	.25	.25	.25	.25	1	1	1	1
Enable remote logging - machine Gabriel	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Enable remote logging - machine Limbo	.75	1	.75	.75	1	1	.25	.25	.25	.25	.75	1	1	1
Restrict user activity implementation	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Shutdown host implementation	.25	.5	.25	.5	.5	.75	.25	.25	.75	1	.25	.5	1	1
Suspend user jobs implementation	1	1	1	1	.5	.25	.75	.5	.25	.25	.75	.5	.25	.25
System Compromise Attack automountd	.25	.5	.25	.5	.75	1	.25	.75	1	1	.25	.5	1	1
System Compromise Attack ping	.25	.5	.25	.5	.75	1	.25	.75	1	1	.25	.5	1	1
System Compromise Attack GetAdmin	.25	.5	.25	.5	.75	1	.25	.75	1	1	.25	.5	1	1
Terminate user session implementation	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PTI	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)
Trace the connection - Reverse DNS Lookup	.5	1	.5	.75	.75	1	1	1	1	1	.5	.75	1	1
Trace the connection -Agent-based approach	.5	1	.5	.75	.75	1	1	1	1	1	.5	.75	1	1
Turn off modems implementation	.25	.5	.25	.75	1	1	.25	.5	1	1	.25	.5	1	1
Warn the intruder -Email	1	.5	1	.75	.5	.5	1	1	.5	.5	1	1	.5	.5
Warn the intruder -Talk	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(A) Confidential User Information
(B) Confidential System Information
(C) Confidential User Account
(D) Confidential Special Account
(E) Confidential Client Root
(F) Confidential Server Root
(G) Denial of Service - User
(H) Denial if Service - Group
(I) Denial of Service - All Users
(J) Denial of Service - Transmitted
(K) Integrity User
(L) Integrity Group
(M)Integrity All
(N) Integrity Transmitted

APPENDIX I

ATTACK IMPLICATION CLASSIFICATION MATRIX

The attack implication classification table lists the effect of the attack classification on the formation of a tentative plan. The system attack implication is set by the system administrator and is an evaluative criterion. Cells contain a value between 0.10 and 1.00 representing the importance of a PTI against a particular attack implication level. Higher values indicate more appropriate PTI. The reasoning employed in the formulation of this table is as follows.

- Plan Step level:
  - The plan steps maintain the same weight across implication levels with the exception of the attack back and preserve evidence plan steps. Attack back is only viable at the higher levels of attack implications. Preserve evidence is less important at the lower levels of attack implications.
  - There is a preference to stop attacks as opposed to slowing an attack across all attack implication levels.
- Tactic and implementation levels: The logic employed at these levels is consistent with the logic articulated for the plan step level.

PTI	Low	Medium	High	Critical
Plan Step				
Gather Evidence	-	-	-	-
Preserve Evidence	.5	.5	1	1
Communicate with the attacker	.33	.33	.33	.33
Slow the attack	.5	.5	.5	.5
Stop the attack	1	1	1	1
Identify potential damaged files	-	-	-	-
Protect critical system files	-	-	-	-
Notify the System Administrator	-	-	-	-
Attack the attacking system	.1	.2	.5	1
Social Engineering	.33	.33	.33	.33
Tactics				
Enable additional logging	-	-	-	-
Enable remote logging	-	-	-	-
Enable logging to unchangeable media	.25	.75	1	1
Enable process accounting	-	-	-	-
Enable additional IDS	-	-	-	-
Trace the connection	.25	1	1	1
Employ honey-pot	.25	1	1	1
Employ smoke-pot	.25	1	1	1
Contact the ISP	.25	.5	.75	1
Warn the intruder	-	-	-	-
Force additional authentication	-	-	-	-
Restrict user activity	-	-	-	-
Turn off modems	.25	.5	1	1
Lock user account	-	-	-	-
Suspend user jobs	-	-	-	-
Terminate user session	-	-	-	-
Block IP address	-	-	-	-
Disable the attacked ports or services	-	-	-	-
Shutdown host	.25	.5	.75	1
Disconnect from the network	.25	.5	.75	1
Create backups	-	-	-	-
Employ temporary shadow files	-	-	-	-
Generate a report	-	-	-	-
Generate an alarm	.5	1	1	1
Denial of service attack	.25	.25	.75	1
System compromise attack	.25	.25	.75	1
Implementations				
Enable additional logging implementation	-	-	-	-

PTI	Low	Medium	High	Critical
Block IP address - At the host	-	-	-	-
Block IP address - At the router	-	-	-	-
Contact the ISP by Email	.25	.5	.75	1
Create backups - Complete system	.25	1	1	1
Create backups - Critical system files	-	-	-	-
Denial of service attack SMURF	.25	.25	.75	1
Denial of service attack Fraggle	.25	.25	.75	1
Denial of service attack Tribe Flood	.25	.25	.75	1
Disable the attacked ports or services - All	-	-	-	-
Disable the attacked ports or services - Only the attacker	-	-	-	-
Disconnect from the network implementation	.25	.5	1	1
Employ temporary shadow files implementation	-	-	-	-
Employ honey-pot 1	.25	1	1	1
Employ honey-pot 2	.25	1	1	1
Employ smoke-pot 1	.25	1	1	1
Employ smoke-pot 2	.25	1	1	1
Enable an additional IDS - Black	-	-	-	-
Enable an additional IDS - Gold	-	-	-	-
Force additional authentication - ask user name and password	-	-	-	-
Force additional authentication - ask secret phrase	-	-	-	-
Generate an incident report	-	-	-	-
Generate an alarm - email	-	-	-	-
Generate an alarm - pager	.5	1	1	1
Generate an alarm - speaker announcement	.5	1	1	1
Lock user account implementation	-	-	-	-
Enable logging to unchangeable media - Printer	.25	.75	1	1
Enable logging to unchangeable media - CD-ROM	.25	.75	1	1
Enable process accounting implementation	-	-	-	-
Enable remote logging - machine Gabriel	-	-	-	-
Enable remote logging - machine Limbo	-	-	-	-

PTI	Low	Medium	High	Critical
Restrict user activity implementation	-	-	-	-
Shutdown host implementation	.25	.5	1	1
Suspend user jobs implementation	-	-	-	-
System Compromise Attack automountd	.25	.25	.75	1
System Compromise Attack ping	.25	.25	.75	1
System Compromise Attack GetAdmin	.25	.25	.75	1
Terminate user session implementation	-	-	-	-
Trace the connection -Reverse DNS Lookup	.25	1	1	1
Trace the connection -Agent-based approach	.25	1	1	1
Turn off modems implementation	.25	.5	1	1
Warn the intruder -Email	-	-	-	-
Warn the intruder -Talk	-	-	-	-

APPENDIX J

CD-ROM INSTRUCTIONS

The attached CD-ROM contains the source code of the AAIRS prototype.

Clicking on any of the JAVA source files in the root directory will launch the prototype

VITA

Curtis A. Carver Jr. was born 22 December 1960 in Savannah, Georgia. He is a 1983 graduate of the United States Military Academy, West Point, where he earned the Bachelor of Science degree in computer science and was commissioned as a signal officer in the U.S. Army. He has served in a number of positions in the Army, achieving the rank of Lieutenant Colonel. During that period, he earned the Master of Computer Science degree from Texas A&M University in 1993. In addition to more traditional military assignments, he served as Assistant Professor in the Department of Computer Science at the United States Military Academy and the principal investigator of the Hypermedia Research Group. In 2001 he was awarded a Ph.D. degree in computer science at Texas A&M University in preparation for a faculty position at West Point.

Curt's permanent address is 741 Wilmington Island Road, Savannah, GA 31410.